A Protocol for Systematic Literature Review on Methods to handle Multiple Concerns in Architecture-Based Self-Adaptive Systems

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 new stakeholders' requirements, changes in requirements priorities, and changes in the environments. As the complexity of these software systems increases, so does their supervisions overhead. Therefore, decreasing 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 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). We will further discuss uncertainty, its characteristics, and current techniques to address this challenge in **Error! Reference source not found.**.

A common approach for handling self-adaptation in domain of autonomous and self-adaptive systems is use of feedback loops based on MAPE-K model. MAPE-K model, which is widely used in design of self-adaptive systems, was first introduced by IBM in their white paper (IBM, 2005). Four components (i.e. Monitor, Analyze, Plan, and Execute) of MAPE-K model communicate and collaborate together, and exchange data through Knowledge component to provide the feedback loop functionality.

One well-known framework for self-adaptive systems based on feedback loop is Rainbow model; by adopting an architecture-based approach, the Rainbow framework not only provides a reusable infrastructure, but also helps to expose the systems runtime behavior and properties (Cheng, Garlan, & Schmerl, 2006). 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. Although the Rainbow framework presents an architectural solution to achieve self-adaptation at low cost, it is based on several fundamental technical assumptions regarding the target system. Furthermore, the primary aim of this framework is to support adaptation within a single Rainbow instance in a centralized setup which may not be feasible all the time (Weyns et al., 2012).

When the number of concern[2]s for adaptation grows, so does the complexity of decision making process. Although different techniques (e.g. utility functions) combined with architecture-based self-adaptive methods have been used to handle adaptations in presence of multiple concerns (Cheng et al., 2006), finding best way to handle failure during adaptation execution and preemption (i.e. to recognize and handle time-critical adaptations (Raheja, Cheng, Garlan, & Schmerl, 2010)) remains unsolved.

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

[2] In this document the term "concern" refers to any type of system goal (functional, non-functional) which should be achieved.

In order to develop models that capture the required knowledge of the concerns of interest, and to investigate how these models can be employed at runtime, we need to examine current self-adaptive methods. Therefore, in this study we aim to identify, and summarize current methods handling multiple concerns in self-adaptive systems. Thus, we conduct a systematic literature review, which is a well-defined method to identify and evaluate studies in a specific domain regarding a particular set of research questions (Kitchenham & Charters, 2007). By performing a systematic literature review, we obtain a fair and unbiased evaluation of selected topic using a rigorous and reliable method.

1. Background

In this section we give an overview of the self-adaptive systems domain, their main requirements, and different dimensions to design and implement such systems. Furthermore, we list quality attributes (QAs) descriptions that will potentially be found in data extraction phase.

1.1. Self-Adaptive systems

A self-adaptive system is capable of modifying its structure or behavior for various reasons. A change in the system's environment, system faults, the need for new requirements, and changes in the priority of requirements are considered known reasons for triggering adaptation. A self-adaptive system continuously monitors itself, gathers data, analyzes them and decides if adaption is required. The challenging aspect of designing and implementing a self-adaptive system is that not only must the system apply the changes at runtime, but also fulfill the systems requirements up to a satisfying level, and overcome uncertainty and its impacts on self-additive's ability to achieve system's objectives.

M.Salehie et al. present a taxonomy for self-adaptive systems based on adaptation concerns which are characterized using six factors (Salehie & Tahvildari, 2009). The taxonomy consists of a list of six essential questions which should be addressed regarding self-adaptive systems. This list helps to identify the important requirements needed for designing and developing self-adaptive systems: (1) Where in the system should the change be applied, and in which level of granularity do the components need to be modified? This question aims to locate the problem and collect data on systems behavior and components' dependency and interaction. (2) In what manner should the modification be applied? This question identifies the best timing for application of the change, and to recognize whether or not the changes in the system should be applied continuously. (3) What artifacts should be modified? This question identifies a variety of parameters (e.g., system components, attributes, resources, and architectural styles) which may be adapted in the system. (4) What is the purpose of adaptation? This question addresses goals that the self-adapting system should achieve. The goals are system requirements such as robustness, openness, etc. (5) To what extend should the system be automatically adaptable? This question addresses the level of system automation, and human involvement. (6) How should the change be applied on the particular artifact? This question determines the most appropriate adaptation action, the order of changes, and also addresses the effects of changes on cost and other parameters. M.Salehie et al. state that answering these questions helps designers to extract important requirements in the development phase, and establish alternative solutions in the operating phase.

J.Andersson et al. propose a classification of modeling dimensions for self-adaptive systems. In their classification there are four key modeling dimensions for self-adaptive software systems (Andersson & Lemos, 2009). Each of these dimensions has an associated classification clarifying different characteristics the dimension may have through the lifetime of the self-adaptive system. The first dimension includes those aspects of the system dealing with self-adaptivity of system goals (i.e. objectives that system should achieve). In all circumstances the self-adaptive system should achieve the system goals. The second dimension is associated with origins (i.e., changes) of self-adaptivity which trigger the need for adaptation in the system. The third dimension includes mechanisms used for self- adaptivity and adaptation process, and finally, the fourth dimension relates to the impacts of the self-adaptation mechanism on the system.

These classifications can be used as a common vocabulary among engineers in a design and development team of a self-adaptive system, or for better understanding of important aspects of an existing software system for reverse-engineering purposes.

1.2. Quality attributes in self-adaptive systems

One of the primary outcomes of this systematic literature review is a list of most significantly addressed QAs in domain of self-adaptive systems. However, to avoid ambiguity, 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 these 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 consist 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.).

We do not limit the included primary studies to those 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.

1.3. Problem statement and Justification

The objective of this study is to explore current architecture-based methods handling multiple concerns in domain of self-adaptive systems. As observed by Weyns et al (Weyns & Ahmad, 2013) handling multiple concerns in domain of self-adaptive systems is a neglected research area. Their results indicate that less than half of the existing research in the domain of self- adaptive systems considers multiple concerns in their proposed methods, but little attention is given to the interplay of the concerns. In addition, most studies addressing multiple concerns only discuss the positive effect of self-adaptation on a particular concern (e.g., QA), and probable negative impacts of self-adaptation on the rest of concerns in the system are ignored. Therefore, we conduct this systematic literature review to meticulously investigate the characteristics of existing methods handling multiple concerns. In this document "handling" refers to a) fulfilling multiple requirements through self-adaptation in order to achieve goals of the system, or b) fulfilling a requirement(s) and investigating effects of self-adaptation on a set of quality attributes of the system (i.e. trade-off analysis). The results of this review give an in-depth overview of current methods, help to identify areas for improvement, and serve as a basis for proposing better solutions in the future.

2. Research Methodology: Systematic Literature Review

In this study we aim at identifying, examining, and summarizing current methods handling multiple concerns in self-adaptive systems. Therefore, we conduct a systematic literature review, which is a well-defined method to identify and evaluate studies in a specific domain regarding a set of particular research questions (Kitchenham & Charters, 2007). By performing a systematic literature review, we get a fair and unbiased evaluation of selected topic using a rigorous and reliable method.

An important step of conducting a systematic literature review is to create a protocol. By specifying all the steps performed in the systematic review, the protocol defines how the review is conducted. More specifically, the protocol contains the research questions, search strategy to identify and collect relevant papers, inclusion and exclusion criteria for filtering out irrelevant papers, methodology for extracting data and synthesizing them to answer research questions.

2.1. Research Questions

The main goal of this systematic literature review is to collect data from the selected primary studies and analyze the extracted data to answer the following five research questions.

1. What are the characteristics of the existing architectural methods for handling multiple quality attributes in self-adaptive software systems?
    - 1.1. Which QAs are addressed by existing methods?
    - 1.2. What artifacts are produced by these methods?
    - 1.3. How many feedback loops are used in the primary studies and how are they arranged?
    - 1.4. Is the method centralized or decentralized?
    - 1.5. What are the application domains of the proposed methods?
    - 1.6. What tools/languages are used/developed in these methods?
2. How do these methods handle quality attributes?
    - 2.1. What models are used to represent the requirement characteristics and how are requirements measured?
    - 2.2. What prioritization mechanisms are used to analyze quality attributes in order to choose the best self-adaptation configuration3?
    - 2.3. Does the method provide any evidence for supporting a satisfying level of requirements after adaptation (i.e., assurances)?
3. What tactics/strategies4 are used to realize adaptation in the system at runtime?
4. What are the limitations and strengths of proposed methods?
5. What evidence is available to adopt methods for handling quality attributes in self-adaptive systems?

We pose research question one to study current methods for handling quality attributes in architecture-based self-adaptation, and to get an overview of what requirements are addressed by

3 Self-adaptation configuration refers to the configuration of the system (i.e., selected alternatives) after adaptation.

4 Similar to (Cheng et al., 2006) tactics correspond to a set of factors: the conditions of applicability, a sequence of actions, and a set of intended effects after execution of adaptation on the system. Furthermore, a flow of actions with intermediate decision points to fix a type of system problems is considered as a strategy.

the existing methods. Furthermore, we collect information about the artifacts created while fulfilling the self-adaptation methods, the methods' application domains, their feedback loop mechanisms and the corresponding tooling.

Research question two helps to scrutinize these methods with an emphasis on requirements. We are interested to investigate how the requirements are affected by the application of self-adaptation methods. In other words, we are interested in determining whether requirements are intentionally affected by the self-adaption method to achieve system's goals or affected as a side effect of applying the self-adaptation method while addressing other requirements. Furthermore, we are interested in identifying models, which were used to represent the requirements in order to measure and update them, and whether there is evidence for guarantees of systems' requirements after adaptation (i.e. assurances). In addition, we want to find out what mechanisms are used by these methods to balance the requirements in self-adaptation process. To be more specific, we are interested to figure out how the methods decide which requirements should be taken into account in order to choose the best-suited self-adaptation configuration in different circumstances.

We pose research question three to investigate the main activities taking place by the adaptation module (i.e., managing system) to apply changes on the system.

We pose research question four to recognize the limitations and strong points of current methods and to identify areas for improvement. The answer to this question may be useful for both researchers and practitioners in the domain of self-adaptive systems.

Finally, by answering research question five, we aim to evaluate the quality of the primary study reporting. We assess each of the included primary studies against a set of questions presented in 3.3, and then, a final quality assessment score will be assigned to each of the primary studies. These evaluations are indications of the quality of the primary studies and reporting of the methods.

2.2. Search strategy

The search strategy is used to search for primary studies and is based on:

a. Preliminary searches to identify existing systematic reviews and assessing potential relevant studies,
b. Trial searches and piloting with different combinations of search terms derived from the research questions,
c. Reviews of research results, and
d. Consultation with experts in the field.

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)
	ACM SIGSOFT international symposium on Foundations of software engineering (SIGSOFT)
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)
	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[5] to find more relevant venues in domain of self-adaptive systems, and software architectures and included them in the study.

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

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

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 the final phase we collect and read the papers, and extract data for data analysis.

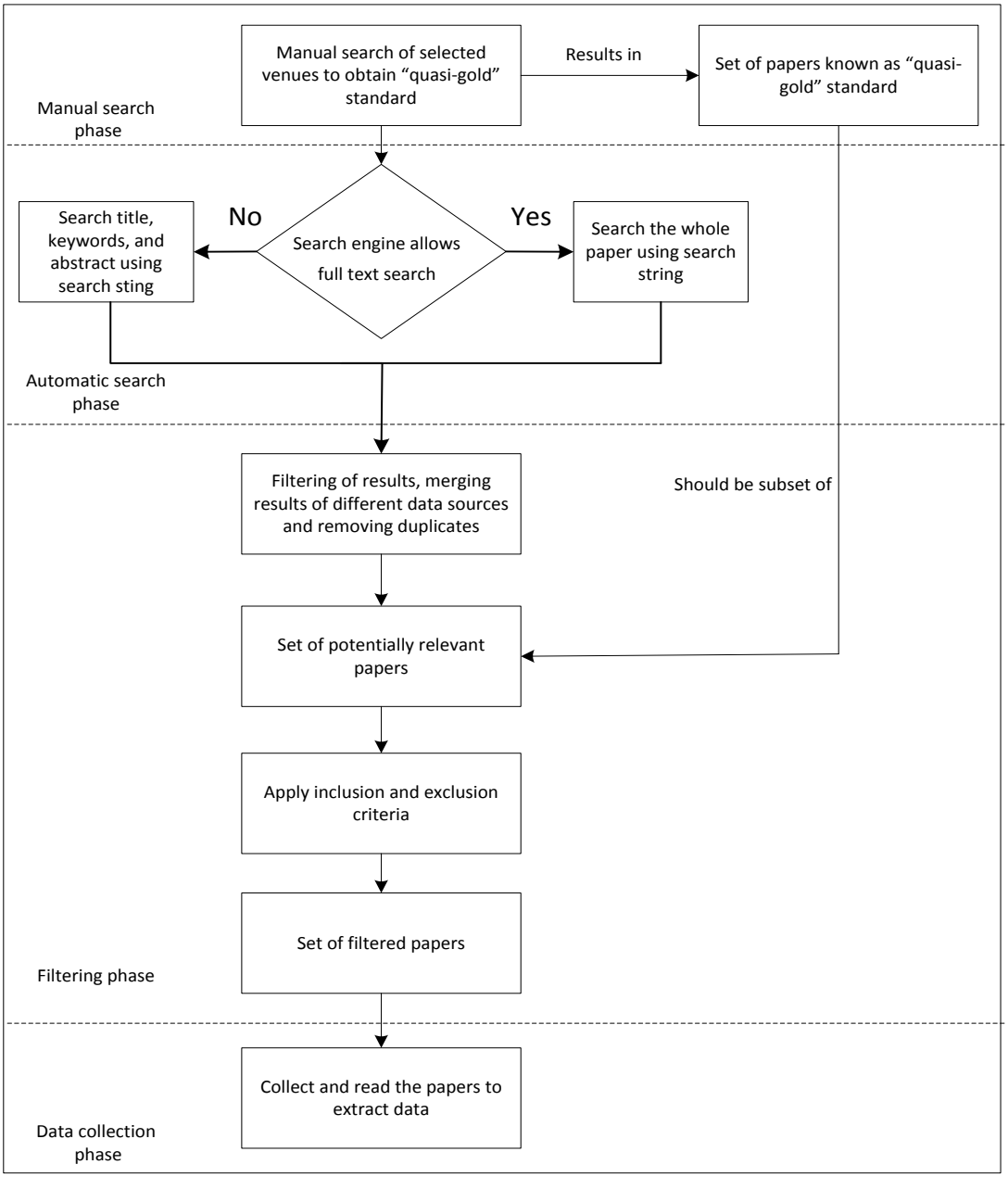

Figure 1- Search process

2.2.5. Inclusion and exclusion criteria

Papers need to cover all the inclusion criteria to be reviewed for the SLR:

Inclusion criteria

1. The study should be in the domain of self-adaptive systems. That is, the system should have a model of itself, and should be able to monitor and adapt itself in certain circumstances in order to fulfill the systems' requirements.
2. The method that deals with self-adaptation should be architecture-based. This implies that the study should provide architectural solutions (i.e., components and architectural models) to handle and reason about the structure and dynamic behavior of the system. To be more specific, the system should use an architectural model of itself to monitor and adapt its structure or runtime behavior. In other words, it should be possible to map the MAPE-k functionalities to components of the adaptation module (managing system).
3. The method presented in the study should handle multiple requirements. This means that the self-adaptation method deals with multiple requirements, or investigates how self-adaptation for one (or more) requirement(s) may affect other quality properties.
4. The study should address two or more system's non-functional or functional requirements. Regarding non-functional requirements, various "ilities" attributes (e.g., availability, security), and "nesses" attributes (e.g., openness, robustness), and other QAs (e.g., performance, efficiency) may be included.

Papers are excluded if they have an intersection with any of the exclusion criteria presented below:

Exclusion criteria

1. The study is an 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 is not written in English.

2.3. Quality assessment

In this SLR, we use a quality assessment method to assess reporting quality of the selected studies (not the quality of research) all selected primary studies are evaluated through a quality check, and the results from this stage will be used for data synthesis and interpretation of results in later stages. All the primary studies which were included in the SLR go through this quality assessment. Similar to [Sarmad, M., Ali, M., Chen, L., & Stol, K. (2010)], we adopted the quality assessment mechanism used in [Dybå, T., & Dingsøyr, T. (2008)]. To assess the quality of the papers, first each paper is evaluated against a set of questions. Answers to each of the questions can be either "yes", "to some extend" or "no", and then numerical values are assigned to the answers (1 = "yes", 0 = "no", and 0.5 = "to some extent").The final quality score for each primary study is calculated by summing up the scores for all the questions. The questions are as follows:

- **Q1:** Is there a rationale for why the study was undertaken?
- **Q2:** Is there an adequate description of the context (e.g., industry, laboratory setting, products used, etc.) in which the research was carried out?
- **Q3:** Is there a justification and description for the research design?

- **Q4:** Is there a clear statement of findings and has sufficient data been presented to support them?
- **Q5:** Did the researcher critically examine their own role, potential bias and influence during the formulation of research questions and evaluation?
- **Q6:** Do the authors discuss the credibility and limitations of their findings explicitly?

The results of quality assessment criteria are used in synthesis phase and to answer research question six. The scores are used as an indication of primary studies' validity, and may be beneficial for practitioners and researchers in domain of self-adaptive systems.

3. Data Extraction and Synthesis

The selected primary studies should be read in detail to extract the data needed in order to answer the research questions. Several researchers read the selected studies, partially or fully, simultaneously. Disagreements need to be resolved by consensus or by consultation with another expert researcher in study domain. Data will be extracted using a data extraction form indicated in Table 2. The data extraction form is as follows:

Table 2 - Data extraction form.

#	Field	Concern / mapping to research question	Additional comments
F1	Author(s)	Documentation	n/a
F2	Year	Documentation	n/a
F3	Title	Documentation	n/a
F4	Source	Reliability of primary study and documentation	n/a
F5	Keywords	Documentation	n/a
F6	Abstract	Documentation	n/a
F7	Citation count (Google scholar)	Documentation	n/a
F8	Method proposed	1	A short summary of the method.
F9	Managing module components	1	Specific components inside MAPE-k loop. Components may be parts of monitoring analyzing, planning, and effector modules. Human interference may also be included in certain scenarios.
F10	Managed system	1	Component(s) being managed by control modules. This may include domain specific software, and execution environment.
F11	Tools/languages	1.6	
F12	Artifacts	1.2	Any tangible item produced by the proposed method.
F13	Feedback loop organization	1.3	Single, multiple, and application style[6] of the loop in the system (e.g., in layered style).
F14	Centralized or decentralized	1.4	n/a
F15	Domain	1.5	If the domain is unclear, it should be stated as “unclear”. Or, if a toy example, or industrial evaluation of the proposed method is presented, the domain of the example should be stated.
F16	QA/goal or side effect/relevant subsystem	1.1, 2	List of QAs. Specify if the QAs are objectives of adaptation or side effects of adaptation.

[6] Note that in this document “application style” has a broader definition than “architecture style”; it refers to any type of information regarding organization and application of feedback back loops in a system. For example, application style may only refer to use of multiple or single feedback loops, or multiple feedback loops organized in layered format in a system.

F17	Models	2.1	Refers to any type of model at design or runtime that represent a QA.
F18	Requirements selection mechanisms	2.2	n/a
F19	Adaptation strategies/tactics	3	List of activities.
F20	Limitations / Weaknesses/ Strengths	4	A brief description of methods' limitations.
F21	Assurances	2.3	If explicit evidence is provided for satisfaction of requirements after adaptation, we try to map the approaches to those listed in (Weyns et al., 2014).

Data stored in [F1] to [F3], and [F5] to [F7] are only for documentation purposes. In [F4] we store the searched sources (i.e., where the primary study was published) which indicate the reliability of the primary study. The rest of the data fields are used for data analysis and are briefly explained below:

- **Method proposed [F8]:** A summary of the proposed method should be recorded.
- **Managing module [F9]:** refers to the system part(s) responsible for monitoring the system and performing adaptation if needed.
- **Managed module [F10]:** refers to part(s) of the self-adaptive system that need to be monitored and adapted. It may include a variety of different items from parameters and methods to components, architecture style, and system resources (Salehie & Tahvildari, 2009). Parameters or system components are not necessarily required to be affected by adaptation actions in order to be considered as managed modules. All the parameters or system components which are monitored are taken into account as managed modules.
- **Tools/language [F11]:** refers to the existing tools/languages which are used by the proposed method or new tools/languages which are designed and implemented in the primary study.
- **Artifacts [F12]:** any tangible item (e.g., diagrams, models, requirement or design documents, etc.) produced by proposed self-adaptation method is considered to be an artifact.
- **Type of loop [F13]:** refers to the characteristics of the MAPE-K model used in the system design, as well as the format (i.e., single or multiple) of used feedback loop.
- **Centralized or decentralized [F14]:** it addresses the issue of having a single (i.e., centralized) or multiple (i.e., decentralized) control components (i.e., analysis and plan from MAPE-K) for decision making. In a centralized method, one control component realizes the need for adaptation of a software system/component, but in a decentralized setting the system's runtime behavior emanates from the localized decisions made by multiple control components and their interactions (Lemos et al., 2013).
- **Domain [F15]:** application domain of the proposed method.
- **QA [F16]:** refers to any QA handled by the self-adaptation method. It can be one of the goals of the self-adaptation system, or it can be due to the application of self-adaptation method (i.e., side effect of the method). For instance, if a system needs to be scalable (i.e., objective of the system) the proposed method should handle scalability. But, if the performance is affected as a side effect of supporting scalability, it should be recorded as an impact of the proposed method.

- **Models [F17]:** refers to models used to represent requirements and their characteristics, and measurements both at runtime and design time. For instance, if the intended quality attribute is reliability, the characteristics may be fault tolerance, and recoverability, and the measurement indicates the threshold for adaptation. In other words, these models may be used to define when and how the requirements should be updated, and what their dimensions (i.e., characteristics) are.
- **Requirements prioritization mechanisms [F18]:** refers to mechanisms used to select the appropriate (i.e., give preference to the most relevant) requirements in order to achieve system's goals in different circumstances (e.g., utility functions).
- **Adaptation strategies/tactics [F19]:** adopted form (Cheng et al., 2006), refers to a flow of actions over a sizable time frame and scope, with intermediate decision points, to fix a type of system problem and meet quality related issues. Actions may refer to any step of decision making process, such as selecting the appropriate data to be collected, as well as analysis and planning methods. All these actions should be identified and recorded as elements of adaptation strategies.
- **Limitations / Strengths [F20]:** addresses the methods' technical limitations and discussed in the primary study or the strong points of the proposed method. Strengths may be any of the following options:

    1) Rigorous evaluation methods (e.g. industrial example, case study).
    2) The proposed method is compared with similar existing methods.
    3) The method is not domain specific.
    4) The method is applicable in both centralized and decentralized settings

- **Assurance [F21]:** refers to method's ability to deliver evidence that the system satisfies its stated functional and non-functional requirements during its operation in the presence of self-adaptation (B. H. C. Cheng et al., 2014). If the primary study provides any mechanism to check whether or not the system complies with its functional and non-functional requirements; we use Table 3 to categorize the identified mechanism.

Table 3 - Approaches for assurances.

Group	Approaches
Human-driven approaches	- Formal proof - Simulation
System-driven approaches	- Runtime verification - Sanity checks - Contracts
Hybrid approaches	- Model checking - Testing

Approaches listed in table 3 are adopted from (Weyns et al., 2014). Listed approaches are representatives of the assurances approaches which have been developed through the years. Depending on the nature of the approach (i.e., manual, automatic, and combination of both) approaches are categorized into three different groups (i.e., Human-driven, system-driven, and hybrid approaches).

4. Data analysis and report

Data extracted from the collected studies will be recoded in excel files to be analyzed and synthesized to answer the research questions. Therefore, the extracted data and their assigned comments should be meticulously read and categorized in a tabulated format to be analyzed. We plan to perform descriptive statistics 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. We will use appropriate statistical methods (e.g., mean, standard deviation, correlation, etc.) to reduce large amounts of data to a simpler summary. Combined with plot representations of the analyzed data, the reduced data then can be used for comparisons across all data, and to answer the research questions. Data analysis process includes the following steps:

1. Review of the comments fields, discussion, and reaching consensus among researchers (if applicable).

2. Data classification and preparation for analysis, and mapping of data to research questions.

3. Statistical data analysis to conduct useful information and conclusions.

In the reporting phase we present the following sections:

1. Discuss main findings of review, details of data analysis and conclusions.
2. Extensive answers to research questions, and the importance and usefulness of related findings.
3. Recommendations, open areas, unanswered questions for future research.

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. To mitigate this, we plan to follow two different strategies. First, we design the data extraction form in such a way that we present pre-defined categories/options for each data field (if applicable). This provides the researcher(s) with a guideline for data extraction and avoids confusion and subjective data extraction. However, it might be even difficult to assign data to different data categories in certain cases. Therefore, not only we record the relevant data in data fields, but also we record a very short comment about the data, if necessary. These recorded comments are useful later on in data analysis phase to assure data is used in a meaningful way. The second solution to alleviate bias is 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

Andersson, J., & Lemos, R. De. (2009). Modeling dimensions of self-adaptive software systems. *… for Self-Adaptive Systems*. Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-02161-9_2

Baresi, L., Pasquale, L., & Spoletini, P. (2010). Fuzzy Goals for Requirements-Driven Adaptation. In *Requirements Engineering Conference (RE), 2010 18th IEEE International* (pp. 125–134).

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

Cheng, S.-W., Garlan, D., & Schmerl, B. (2006). Architecture-based self-adaptation in the presence of multiple objectives. *Proceedings of the 2006 International Workshop on Self-Adaptation and Self-Managing Systems - SEAMS '06*, 2. doi:10.1145/1137677.1137679

IBM. (2005). An architectural blueprint for autonomic computing .

Elkhodary, A., Esfahani, N., & Malek, S. (2010). FUSION: a framework for engineering self-tuning self-adaptive software systems. In *Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering* (pp. 7–16). New York, NY, USA: ACM. doi:10.1145/1882291.1882296

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

Hofmeister, C., Kruchten, P., Nord, R. L., Obbink, H., Ran, a., & America, P. (2005). Generalizing a Model of Software Architecture Design from Five Industrial Approaches. *5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05)*, 77–88. doi:10.1109/WICSA.2005.36

Hutchison, D., & Mitchell, J. C. (n.d.). *Lecture Notes in Computer Science*.

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

Lemos, R. De, Giese, H., & Müller, H. (2013). Software engineering for self-adaptive systems: A second research roadmap. *Software Engineering for …*, 1–32. Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-35813-5_1

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

Raheja, R., Cheng, S., Garlan, D., & Schmerl, B. (2010). *Improving architecture-based self-adaptation using preemption*. Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-14412-7_2

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

Weyns, D. Bencomo, N., Calinescu, R., Camara, J., Ghezzi, C., Grassi V., Grunske, L., Inverardi, P., Jezequel, J-M., Malek, S., Mirandola, R., Mori, M., & Tamburrelli. G. Perpetual assurances in self-adaptive systems. In Assurances for Self-Adaptive Systems, Dagstuhl Seminar 13511, 2014

Weyns, D., Schmerl, B., Grassi, V., Malek, S., Prehofer, C., & Wuttke, J. (2012). On Patterns for Decentralized Control in Self-Adaptive Systems, 76–107.

Whittle, J., Sawyer, P., Bencomo, N., Cheng, B. H. C., & Bruel, J.-M. (2009). RELAX: Incorporating Uncertainty into the Specification of Self-Adaptive Systems. In *Requirements Engineering Conference, 2009. RE '09. 17th IEEE International* (pp. 79–88).

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