A Classification Framework of Uncertainty in Architecture-Based Self-Adaptive Systems with Multiple Quality Requirements

Sara Mahdavi-Hezavehi[1,2], Paris Avgeriou[1], Danny Weyns[2]
[1] Department of Mathematics and Computing Science, University of Groningen, Netherlands
[2] Department of Computer Science, Linnaeus University, Växjö, Sweden
s.mahdavi.hezavehi@rug.nl
paris@cs.rug.nl
danny.weyns@lnu.se

Abstract

Context: The underlying uncertainty in self-adaptive systems aggravates the complexity of selecting best adaptation action alternative, and handling requirements trade-offs. To efficiently tackle uncertainty, it is necessary to have a comprehensive overview of different types of uncertainty and their specifications.

Objective: In this paper we aim at a) reviewing the state-of-the-art of architecture-based approaches tackling uncertainty in self-adaptive systems with multiple quality requirements, b) proposing a classification framework for this domain, and c) classifying the current approaches according to this framework.

Method: We conducted a systematic literature review by performing an automatic search on twenty seven selected venues and books in the domain of self-adaptive systems.

Results: We propose a classification framework for uncertainty and its sources in the domain of architecture-based self-adaptive systems with multiple quality requirements. We map 51 identified primary studies into the framework and present the classified results.

Conclusions: Our results help researchers to understand the current state of research regarding uncertainty in architecture-based self-adaptive systems with multiple concerns, and identity areas for improvement in the future.

1. Introduction

Software systems are subject to continuous changes due to new requirements and the dynamics of the system context. Engineering such complex systems is often difficult as the available knowledge at design time is not adequate to anticipate all the runtime conditions. Missing or inaccurate knowledge may be due to different types of uncertainty such as vagueness regarding the availability of resources, operating conditions that the system will encounter at runtime, and the emergence of new requirements while the system is operating. We define, uncertainty in a software system as the circumstances in which the system's behavior deviates from expectations due to dynamicity and unpredictability of a variety of factors existing in software systems.

One way to deal with this uncertainty is to design systems that adapt themselves during runtime, when the knowledge is accessible. Self-adaptive systems are capable of autonomously modifying their runtime behavior to deal with dynamic system context, and changing or new system requirements in order to provide dependable, and recoverable systems (Lemos et al., 2013). In this research, we focus on architecture-based approaches (Oreizy et al. 1998, Garlan et al. 2004, Kramer and Magee 2007), which are widely used to support self-adaptation. Architecture-based self-adaptive systems achieve this capability by means of using reflective software architecture models. In order to manage a system, an architecture-based self-adaptive system is equipped with adaptation software that uses models of the system, its environment, and goals when monitoring the running system, to detect problems, identify solutions, and apply adaptation action to modify the system.

However, incorporating self-adaptation into a system may lead to further uncertainty in its own right: Defective adaptation actions or unforeseen consequences of adaptation on the system can result in unexpected system behavior. This is further aggravated in the case of self-adaptive systems that need to simultaneously fulfill multiple quality requirements without interrupting the system's normal functions, and deal with a growing number of both adaptation scenarios and requirements trade-offs (Cheng et al., 2006). 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. However, when the number of system quality requirements increases, so does the number of adaptation alternatives. Therefore, the decision making, as well as the handling of requirements trade-offs becomes more complex. If the problem is not handled properly, over time uncertainty provokes inconsistency in certain subsystems, and the accumulated inconsistencies may result in unforeseen circumstances, and possibly in unexpected system behavior.

Over the past years, numerous approaches have been proposed to quantify and mitigate existing uncertainty in self-adaptive systems. However, the concept of uncertainty and its different types and categories are hardly ever studied in the domain of architecture-based self-adaptive systems with *multiple quality requirements*. As a result, identification, investigation, and consequently selection of suitable approaches for tackling uncertainty in this domain may be problematic. To alleviate this problem, in this paper we present a framework to classify existing uncertainty concepts for architecture-based solutions in self-adaptive systems with multiple quality requirements. To create the framework, we systematically review all the papers that propose approaches to deal with uncertainty and its sources. Subsequently we study these approaches according to the proposed classification framework in order to facilitate their potential comparison and selection. This classification framework may further be used to propose new solutions tackling the uncertainty problem more efficiently in the future.

This paper is organized as follows: in Section 1 we present background and related work. In Section 2 we introduce our research questions, discuss both the search strategy, and data extraction method. In Section 3 we present the results of the study, and extensively answer the research questions. In Section 4 we discuss the results of the study including main findings, limitations of the study, and threats to validity. Finally, Section 5 concludes the paper.

1.1. Background

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

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.

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. The essential functions of architecture-based self-adaptation are defined in the MAPE-K (i.e., Monitor, Analyze, Plan, Execute, and Knowledge component) reference model (IBM, 2005). By complying with the 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 (Weyns et al., 2012). 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) (Walsh et al, 2004) 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 a 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, when the number of adaptation dimensions increases, representing the choices for adaptation, and updating and maintaining trade-offs becomes problematic

(Cheng et al., 2006). Therefore, the majority of current architecture-based self-adaptive systems approaches do not address trade-offs analysis explicitly, and specifically the negative impacts of the applied adaptation method on multiple quality attributes, which deteriorates systems' overall quality in complex software systems. A recent survey (Danny Weyns & Ahmad, 2013) summarizes the state of the art in architecture-based adaptation in general, and handling multiple requirements in particular.

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. The first and foremost genre of uncertainty is the 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 a solution space, choose the optimal adaptation action, and adapt the system while fulfilling quality requirement of the system in a specified 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 a 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.

1.2. Related work

During the past decade several studies have been conducted to address uncertainty issue in different phases of software systems life cycle. (Rotmans et al., 2003) attempt to harmonize the uncertainty terminology by proposing a conceptual framework (i.e., uncertainty matrix which considers uncertainty from three different dimensions: location, level of uncertainty, and nature of uncertainty), which helps to identify and characterize uncertainty in model-based decision support activities. Although the uncertainty matrix presented in that paper can be used as a guideline in the domain of self-adaptive systems as well; we found it difficult to use their detailed taxonomies and definitions of uncertainty dimensions, as it is mainly applicable to the field of model-based decision support. Following the same theme of uncertainty dimensions (i.e., location, level, and nature of uncertainty), (Perez-palacin et al., 2014) present a taxonomy for uncertainty in the modeling of self-adaptive systems. In their work, they also provide an extensive list of examples for sources of uncertainty, which is extracted from the literature. Nonetheless, the authors do not manage to provide descriptions for the sources of uncertainty. In (Refsgaard et al., 2007), the authors present terminology and a topology of uncertainty and explore the role of uncertainty at different stages of a water management modeling process. However, their terminology is substantially inspired by work of (Rotmans et al., 2003), and their field of research is remarkably different from our domain of interest; which makes it difficult to apply their work in the domain of self-adaptive systems. In (David Garlan, 2010), the author argues that in today's software systems uncertainty should be considered as a first-class concern throughout the whole system life cycle, and discusses a number of sources of uncertainty affecting software systems. What we think is missing in this work is the mapping of these sources of uncertainty into the

previously discussed dimensions and taxonomies of uncertainty in the literature. (Esfahani & Malek, 2013) mostly focus on sources of uncertainty, and present an extensive list of sources with examples. Moreover, they investigate uncertainty characteristics (reducibility versus irreducibility, variability versus lack of knowledge, and spectrum of uncertainty), and sources of uncertainty characteristics in their work; however the connection between these characteristics and dimensions of uncertainty is unclear. Lastly, (Ramirez, Jensen, & Cheng, 2012) provide a definition and taxonomy for uncertainty in dynamically adaptive systems. The presented taxonomy describes common sources of uncertainty and their effect on requirements, design and runtime phases of dynamically adaptive systems. The main focus of this paper is sources of uncertainty as well.

Investigating the current state of research regarding uncertainty in software systems, and identifying gaps and inconsistencies in the literature motivated us to conduct an exhaustive review of the topic in domain of architecture-based self-adaptive systems with multiple quality requirements. We argue that it is crucial to systematically study and grasp current approaches, investigate different dimensions of uncertainty to precisely comprehend the problem statement (i.e., uncertainty definition, dimensions, sources, etc.), and to identify issues which need to be resolved in order to propose approaches that can be tailored and reused in a variety of systems. The classification framework we present aims to provide a consistent and comprehensive overview of uncertainty and its specifications in domain of architecture-based self-adaptive systems with multiple quality requirements.

2. Study Design

In this study we aim at identifying, exploring, and classifying the 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[1], in which all the steps and details of the study are specified. In this section, we report parts of the protocol and its execution: 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 the data synthesis method we used 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 software system that is subject of adaption (i.e., the managed system) should be equipped with adaptation software that uses architectural models of the system, its environment, and goals when monitoring the running system and adapt the managed system at runtime when needed. In particular, it should be possible to map the components of the adaptation software to MAPE-k functionalities. With multiple requirements, we refer both to approaches that handle more than one adaptation concern (e.g., adapt for reliability and security) and approaches that consider a single adaptation concern (e.g., reliability) but also the effects on one or more other concerns (e.g., performance overhead). The 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 aims 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, the managed system, components of the adaptation software) 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

[1] The protocol is available at: http://www.cs.rug.nl/search/uploads/Resources/book_chapter_protocol.pdf

discussed in the selected papers or not. The 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 aims to understand the dimensions of uncertainty resulting from answering 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 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 aim 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 and three 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 the domain of self-adaptive systems with multiple quality requirements. Essentially, we 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 aim to 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

In this section, we present the main steps we performed in order to identify, filter, and include all the relevant papers in our study. An extended and more detailed description of our search strategy can be found in the protocol.

2.2.1. Search scope and automatic search

The scope of the search is defined in two dimensions: publication period 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. We chose this start date because the development of successful self-adaptive software hardly goes back to a decade ago; after the advent of autonomic computing (Kephart & Chess, 2003). Note that even though some major venues on self-adaptive systems started to emerge after 2005 (e.g., International Symposium on Software Engineering for Adaptive and Self-Managing Systems), we chose to start the search in the year 2000 to avoid missing any studies published in other venues.
Since the number of published papers in this domain is over several thousand, manual search was not a feasible approach to search databases (Ali et al., 2010). Therefore, we used the automatic search method to

search through selected venues. By automatic search we mean search performed by executing search strings on search engines of electronic data sources (i.e., IEEE Xplorer, ACM digital library, SpringerLink, and ScienceDirect). An advantage of automatic search is that it supports easy replication of the study.
One of the main challenges of performing an automatic search to find relevant studies in the domain of self-adaptive systems was 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 decided to use a more generic search string and include a wider number of papers in the initial results. We used the research questions and a stepwise strategy to obtain the search terms; the strategy is as follows:

1) Derive main terms from the research questions and the topics being researched.
2) If applicable, identify and include alternative spellings and synonyms for the 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 meaningful combinations of search terms.
5) Check the pilot results with the "quasi-gold" standard which is a set of manually derived primary studies from a given set of studies (see below for further explanation).
6) Organized discussions between researchers to adjust the search terms, if necessary.

As a result, the following terms were 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 based on the combination of key terms: 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)

Although manual search is not feasible for databases where the number of published papers can be enormous, we still incorporated a manual search (i.e., "quasi-gold" standard (Zhang & Ali Babar, 2010)) into the search process to make sure that the search string works properly. To establish the "quasi-gold" standard, we manually searched three different venues. To perform the manual search, we looked into papers' titles, keywords, abstracts, introductions, and conclusions. The manually selected papers were cross-checked with the results of automatic search to ensure that all the relevant papers are found during the automatic search. This means that papers found for "quasi-gold" standard should be a subset of automatic results. This step (i.e., creating "quasi-gold" standard) ensures validity of the created search string.

In total, we have selected and included 51 papers derived from 27 different venues and books. To be more specific, the venues include 13 different conferences, 4 workshops, 7 journals, and 3 books.

2.2.2. Overview of Search Process

We adopted a four phased search process to search the selected venues and books, filter results, and collect relevant papers. The different steps of the process are shown in Figure 1.

Figure 1- Search process.

In the first phase (i.e., manual search), we manually searched three selected venues (see Table 18) to create the "quasi-gold" standard. The final set of papers from this phase should be cross checked with the automatic results in the filtering phase. In the next phase (i.e., automatic search), we performed the automatic search of selected venues (see Table 19). Depending on the search engines' capabilities, different search strategies were picked. If the search engine allowed, we used the search string to search the full paper; otherwise, titles, abstracts and keywords were searched. In the filtering phase, we filtered the results based on titles, abstracts, keywords, introductions, and conclusions, and also removed the duplicate papers.

We ended up having 7453 potentially relevant papers which then were compared with the "quasi-gold" standard. Since the "quasi-gold" standard papers were a subset of potentially relevant papers, we proceeded to the next step and started filtering the papers based on inclusion and exclusion criteria. At this point, we started reading the whole papers as it was not possible to filter some of the papers only based on abstract, introduction, and conclusion. Therefore, for certain papers we also started extracting and collecting data simultaneously. Finally, we included 51 papers as our primary studies, and finished the data extraction for all of the papers.

2.2.3. Refining the Search Results

We used the following inclusion and exclusion criteria to filter our extracted set of papers.

2.2.3.1. 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 the systems adaptation logic to MAPE-k functionalities.
3) The study should tackle multiple quality requirements, either as a goal of adaptation or as a consequence of applying a self-adaptation method.

2.2.3.2. 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 written in English.

2.3. Data extraction

We used our selected primary studies to collect data and answer the research questions. Our data extraction approach was semi-structured. We created initial uncertainty dimensions and source classification schemas (see Table 1, and Table 2) based on the literature, namely the work by (Perez-palacin et al., 2014), (Refsgaard et al., 2007), (Rotmans et al., 2003), (David Garlan, 2010), (Esfahani & Malek, 2013), and (Ramirez et al., 2012). Our intent was to extend and complete both the dimension and source classifications schemas based on data we extract from the primary studies.

Table 1 - Uncertainty dimensions initial classification schema.

Uncertainty Dimension	Dimension Descriptions
Location (Walker et al., 2003)	"It is an identification of where uncertainty manifests itself within the whole model complex."
Nature	"Specifies whether the uncertainty is due to the

(Walker et al., 2003)	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 2 - Sources of uncertainty initial classification schema.

Uncertainty Source	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.

We also recorded comments to capture additional observations about certain papers or data fields; the comments were used to solve any disagreements among researchers, if necessary.

2.4. Data items

Table 3 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 3 - 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

2.5. Quality assessment of selected papers

We use a quality assessment (QA) method to assess the quality of all the selected papers that were included in this review. We adopted the quality assessment mechanism (i.e., definitions and quality assessment questions) used by Dyba and Dingsoyr (Dybå & Dingsøyr, 2008) as follows:

- **Quality of reporting:** Papers' rationale, aim, and context should be clarified.
  - **QA1:** Do the authors clarify the aims and objectives of the paper, and is there a clear rationale for why the study is undertaken?
  - **QA2:** Is there an adequate description of the context in which the research was carried out?

- **Rigour:** A thorough and appropriate approach is applied to key research methods in the paper.
    - **QA3:** Is there an adequate justification and clear description for the research design?
- **Credibility:** The papers' findings are well presented and meaningful.
    - **QA4:** Has sufficient data been presented to support the finding, are the findings are stated clearly?
    - **QA5:** Do the researcher examine their own potential bias and influence during the formulation of research questions and evaluation of results?
    - **QA6:** Do the authors discuss the credibility and limitations of their findings?

The quality assessment mechanism of Dyba and Dingsoyr covers also relevance (i.e., explores the value of the paper for the related community) of papers. However, in this systematic review we have only included papers published in high quality venues that are relevant to our domain of interest, thus further investigation of usefulness of the papers for the community is unnecessary.

To assess the quality of the papers, each paper is evaluated against the abovementioned quality assessment 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 paper is calculated by summing up the scores for all the questions. The results of quality assessment are used in the synthesis phase to support the validity of included papers in this review. The scores assigned to the selected papers are presented in Section 3.1.

3. Results

In this section we present a basic analysis of our results through various tables and charts, and then answer the research questions.

3.1. Quality of selected papers

Our list of venues (Table 19) for automatic search includes the list of venues searched by Weyns at al. in (Weyns & Ahmad, 2013). In that systematic literature review, the authors included a list of high quality primary studies in the domain of self-adaptive systems, software architectures, and software engineering. Furthermore, to broaden the search scope and extend the list of venues, we used Microsoft Academic Search2 to find more relevant venues in the domains of self-adaptive systems and software architecture, and included them in the study. However, to verify the quality of selected papers furthermore, we assessed all the papers based on the method described in section 2.5. In Figure 2 we indicate all the selected papers and their associated quality assessment scores. Green bubbles contain papers with average quality scores (i.e., scores of 4 and 4.5), and orange bubbles contain papers with higher quality scores (i.e., scores of 5, 5.5, and 6). Blue bubbles contain papers with low quality scores. The results suggest that the selected papers for this study are of relatively high quality: 18 papers are located score 4 or 4.5, and 22 papers score from 5 to 6.

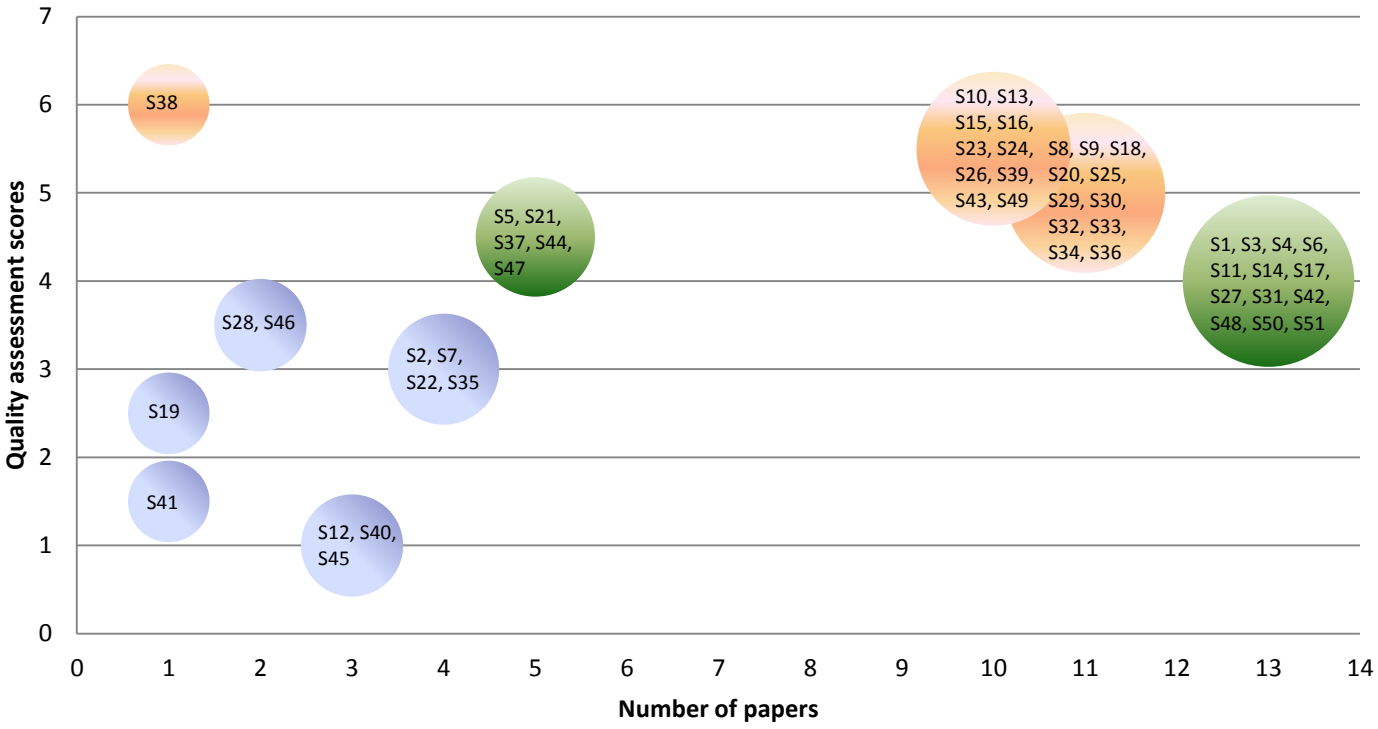

Figure 2 - Quality assessment of selected papers.

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

3.2. RQ1: What are the current architecture-based approaches tackling uncertainty in self-adaptive systems with multiple requirements?

In this study, we included 51 papers in total (see Table 17 for complete list of papers). Figure 3 shows the number of included papers per venue with publication numbers equal or higher than two. Software Engineering for Adaptive and Self-Managing Systems conference (SEAMS) and Software Engineering for Self-Adaptive Systems (SESAS) volumes I and II have the most number of selected papers with 14 publications and 6 papers respectively.

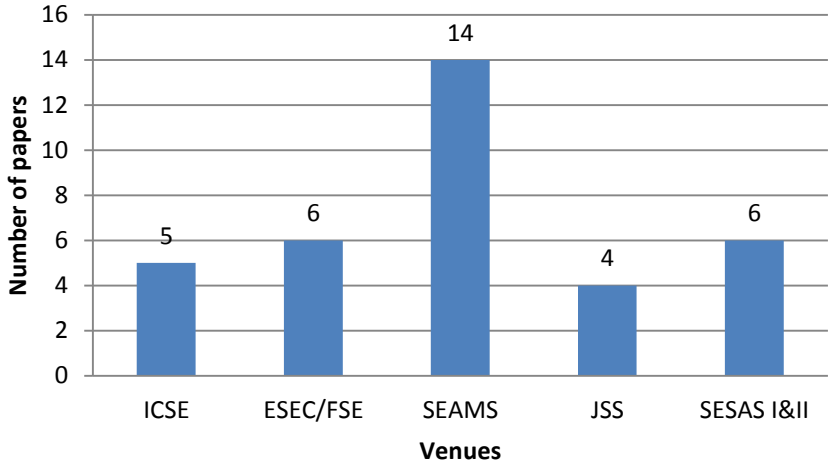

Figure 3 - Number of published papers per venue.

From Figure 4 we can see that most of the studies started to appear around 2009; suggesting that architecture-based approaches tackling uncertainty in self-adaptive systems with multiple quality requirements were not widely studied before the year 2008. Since architecture-based approaches have been used in the domain of self-adaptive system even before 2009, we speculate that uncertainty in self-adaptive systems with multiple quality requirements has been under-studied before the year 2009.

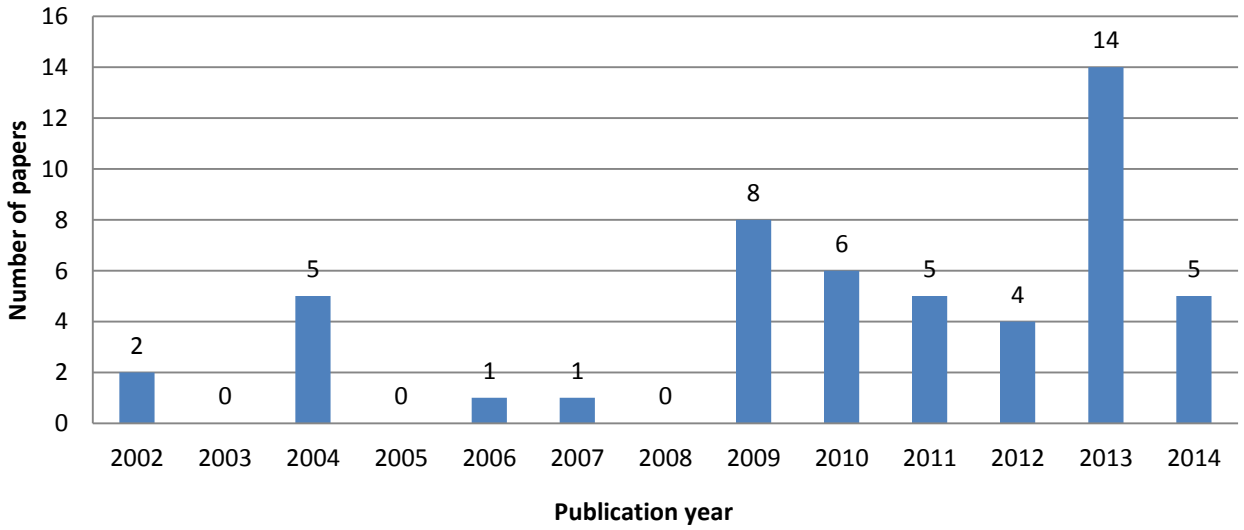

Figure 4 - Number of papers published per year.

3.3. RQ2: What are the different uncertainty dimensions which are explored by these approaches?

We used the initial classification schema of uncertainty dimensions (see Table 1) to extract data from the selected papers, and then gradually extended that initial classification schema to create our framework. Table 4 presents a list of significant dimensions we found in the literature, descriptions of the dimensions, and possible options for each of the dimensions.

Table 4 - Classification framework for dimensions of uncertainty and its options.

Uncertainty Dimension	Description	Options	Descriptions
Location	Refers to the locale, where uncertainty manifests itself within the whole system.	Environment	Refers to execution context and humans interacting with, or affecting the system.
		Model	Refers to a variety of conceptual models representing the system.
		Adaptation functions	Refers to functionalities performed as part of MAPE-K model.
		Goals	Refers to specification, modeling and alteration of system goals.
		Managed system	Refers to the application specific system, which is being monitored and adapted.
		Resources	Refers to a variety of essential factors and components which are required by the self-adaptive system in order to operate normally.
Nature	Specifies whether the uncertainty is due to the imperfection of available knowledge, or is due to the inherent variability of the phenomena being described.	Epistemic	The uncertainty is due to the imperfection of our knowledge, which may be reduced by more research and empirical efforts.
		Variability	The uncertainty is due to inherent variability in the system complex including randomness of nature, human behavior, and technological surprises.
Level /Spectrum	Indicates the position of uncertainty along the spectrum	Statistical uncertainty	Statistical uncertainty refers to deterministic knowledge in the uncertainty spectrum and is any uncertainty that can

	between deterministic knowledge and total ignorance.		be described adequately in statistical terms.
		Scenario uncertainty	A scenario is a plausible description of how the system and or its driving forces may develop in the future. Scenarios do not forecast what will happen in the future; rather they indicate what might happen.
Emerging Time	Refers to time when the existence of uncertainty is acknowledged or uncertainty is appeared during the life cycle of the system.	Run time	Refers to the uncertainties appearing after systems deployment, which also includes system evolution over time.
		Design time	Refers to the uncertainties manifesting themselves during any software development phases carried out before system deployment.
Sources	Refers to a variety of circumstances affecting the adaptation decision, which eventually deviate system's performance from expected behavior		See Table 5

As indicated in Table 4, we found five different noteworthy dimensions of uncertainty(i.e., Location, Nature, Level, Emerging Time, and Sources). This implies that current architecture-based approaches in the domain of self-adaptive systems with multiple quality requirements examine uncertainty from five distinct perspectives. The fact that these dimensions were extracted from the literature suggests that any effective solution tackling uncertainty should at least address these dimensions in order to thoroughly explore underlying uncertainty in self-adaptive systems, and afterwards, propose solutions to tackle uncertainty.

Notice that the primary dimensions descriptions listed in Table 1 were refined into those presented in Table 4. Although undertaking the systematic review did not change the core of the definitions presented in the primary classification schema, it did help to refine the definitons in order to be further applicable in the domain of architecture-based self-adaptive systems and to fit into the final classification framework.

3.3.1. RQ2.a: What are the options for these uncertainty dimensions?

In Table 4, we also provide detailed descriptions for each of the options listed for uncertainty dimensions. Furthermore, we expanded the options list by adding new options (i.e. managed system, and sources) to the primary schema. By providing a full list of options and their descriptions, this table can be used as a guideline for researchers to avoid any ambiguity while addressing dimensions options in their work.
We note that the dimension 'level of uncertainty' may also include recognized ignorance (i.e., acknowldeging uncertainty, but not proposing any remedy), and total ignorance (i.e., completely ignoring the existance of uncertainty) as options. However, these two options do not apply for any of the primary studies: all the studies acknowledge existance of uncertainty and propose solutions to handle it.

3.4. RQ3: What sources of uncertainties are addressed by these approaches?

Finally, to answer this research question, we used the initial classification schema for sources of uncertainty (see Table 2) for data extraction and created an extended list of sources of unecrtainty. In Table 5 we present the extended list, along with the descriptions for the options and examples from literature. The sources of uncertainty refer to a variety of circumstances from which the uncertainty originates. We also added one more column, "classes of uncretainty" which is only used for grouping purposes: sources of uncertainties with similar origins are grouped in the same class of uncertaitny. This helps making a long list of sources of uncertainty easier to analyze in the next section.

Table 5 - Sources of uncertainty.

Class of source of uncertainty	Options (for Sources of uncertainty)	Description	Example
Model uncertainty	Abstraction	Uncertainty caused by omitting certain details and information from models for the sake of simplicity.	Simplifying assumptions (Esfahani & Malek, 2013)
	Incompleteness	Uncertainty caused by parts (of models, mechanisms, etc.) that are knowingly missing because of a lack of (current) knowledge.	Model structural uncertainty (Perez-palacin et al., 2014)
	Model drift	Uncertainty caused by a discrepancy between the state of models and the represented phenomena.	Violation of requirements in models (Carlo Ghezzi & Sharifloo, 2013)
	Different sources of information	Uncertainty caused by differences between the representations of information provided by different sources of information. Uncertainty may be due to different representations of the same information, or result of having different sources of information, or both.	Granularity of models (Cheung et al., 2007)
	Complex models	Uncertainty caused by complexity of runtime models representing managed sub systems.	Complex architectural models (Vogel & Giese, 2010a)
Adaptation functions uncertainty	Variability space of adaptation	Uncertainty caused by the size of the variability space that the adaption functions need to handle. This type of uncertainty arises from striving to capture the whole complex relationship of the system with its changing environment in a few architectural configurations which is inherently difficult and generates the risk of overlooking important environmental states [5].	Being unable to foresee all possible environment states as well as all the system configurations in the future (Chauvel et al., 2013)
	Sensing	Uncertainty caused by sensors which are inherently imperfect.	Noise in sensing (Esfahani & Malek, 2013)
	Effecting	Uncertainty caused by effectors of which the effects may not be completely deterministic.	Futures parameter value (Esfahani & Malek, 2013)
	Automatic learning	Uncertainty caused by machine learning techniques of which the effects may not be completely predictable.	Modeling techniques (Cheung et al., 2007)
	Decentralization	Uncertainty due to decision making by different entities of which the effects may not be completely predictable.	Decentralized control in a traffic jams monitoring system (Weyns et al., 2010)
	Changes in adaptation mechanisms	Uncertainty due to required dynamicity of adaptation infrastructure to maintain its relevance with respect to the changing adaptation goals (Villegas, Tamura, Müller, Duchien, & Casallas, 2013).	Additional monitoring infrastructure (Villegas et al., 2013)
	Fault localization and identification	Uncertainty caused by inaccurate localization and identification of faults in the managed system.	Identifying and ranking faulty component (Casanova et al., 2013)

Goals uncertainty	Goal dependencies	Dependencies between goals, in particular quality goals, may not be captured in a deterministic manner, which causes uncertainty.	Conflict resolution between competing quality attributes (Zoghi et al., 2014)
	Future goal changes	Uncertainty due to potential changes of goals that could not be completely anticipated.	Rapid evolution (David Garlan, 2010)
	Future new goals	Uncertainty due to the potential introduction of new goals that could not be completely anticipated.	Rapid evolution (David Garlan, 2010)
	Goal specification	Uncertainty due to lack of deterministic specifications of quality goals.	Quality goals priorities changes (Esfahani, et al., 2011)
	Outdated goals	Uncertainty caused by overlooking outdated goals.	Addressing goals which are irrelevant to the system (Tamura et al., 2013)
Environment uncertainty	Execution context	Uncertainty caused by the inherent unpredictability of execution contexts.	Mobility (David Garlan, 2010)
	Human in the loop	Uncertainty caused by the inherent unpredictability of human behavior.	Objectives (Esfahani & Malek, 2013)
	Multiple ownership	Uncertainty caused by lack of proper information sharing, conflicting goals, and decision making policies by multiple entities that own parts of the system.	Uncertain execution time and failure rate of a component operated by a third-party organization (C Ghezzi et al., 2013)
Resources uncertainty	New resources	Uncertainty caused by availability of new resources in the system.	Availability of new services in the system (Edwards et al., 2009)
	Changing resources	Uncertainty caused by dynamicity of resources in the system.	Resources mobility (Hallsteinsen et al., 2004)
Managed system uncertainty	System complexity and changes	Uncertainty caused by complexity and dynamicity of nature of the managed system.	Complex systems and complex architectural models (Vogel & Giese, 2010)

In this table, specific examples from the literature are provided to help with the comprehensibility of sources.

3.5. RQ4: How are the current approaches classified according to the proposed uncertainty classification framework?

From 51 selected papers, 12 papers discuss one class of uncertainty. Environment is the most addressed class of uncertainty, and adaptation functions is the least (see Table 6).

Table 6 - List of papers discussing single class of uncertainty.

Class of uncertainty	Number of papers	Study numbers
Environment	4	S20, S34, S37, S38
Goal	3	S4, S29, S41
Model	3	S11, S16, S23
Adaptation functions	2	S5, S14

The rest of the papers (39 out of 51) discuss multiple classes of uncertainty. A variety of combinations of classes of uncertainty are discussed in the literature; "Environment, Goal, and Adaptation functions" is the most addressed set of classes of uncertainty, for details see Table 7.

Table 7 - List of papers discussing combinations of classes of uncertainty.

Classes of uncertainty	Number of papers	Study numbers
Environment, Goal, Adaptation functions	9	S8, S9, S25, S31, S32, S43, S44, S45, S49
Environment, Goal	8	S7, S15, S18, S33, S46, S47, S51
Environment, Adaptation functions	3	S17, S42, S50
Environment, Model, Adaptation functions	3	S13, S12, S19
Environment, Model	2	S3, S24
Environment, Goal, Adaptation function, Model	2	S26, S10
Environment, Goal, Managed system	2	S27, S36
Environment, Goal, Model	2	S30, S40
Adaptation function, Model, Goal	1	S48
Goal, Adaptation function	1	S39
Environment, Resources	1	S2
Environment, Resources, Adaptation functions	1	S21
Environment, Goal, Resources	1	S1
Environment, Adaptation functions, Goal, Managed system	1	S35
Environment, Adaptation functions, Goal, Resources	1	S22
Environment, Model, Managed system	1	S6
Goal, adaptation function, resources	1	S28

From Table 6 and Table 7, we can conclude that the majority of existing studies (i.e., 39 papers) explore different classes of uncertainty, and do not focus on proposing solutions to tackle certain class of uncertainty and its sources. We can also observe that "Environment" and "Goal" seems to be the most important classes of uncertainty, and the majority of researchers are interested in tackling uncertainties emanating from environmental circumstances and self-adaptive system's goal related complications.

Regarding the nature of uncertainty (see Table 8), 35 papers (i.e., 68.6%) discuss uncertainty due to variability, and only two papers tackle uncertainty due to lack of knowledge (i.e., Epistemic). Although 14 papers address both variability and lack of knowledge as the nature of uncertainty in self-adaptive systems; variability seems to be the main source from which uncertainty originates as 35 primary studies 'main focus is only variability.

Table 8 - List of papers and nature of uncertainty.

Nature	Number of papers	Study numbers
Variability	35	S1, S2, S4, S5, S6, S7, S8, S9, S10, S14, S15, S16, S17, S18, S19, S20, S21, S22, S25, S28, S29, S31, S34, S36, S37, S38, S39, S40, S41, S43, S45, S47, S48, S49, S51
Variability, Epistemic	14	S3, S11, S12, S13, S24, S26, S27, S30 , S32, S33, S35, S42, S46, S50
Epistemic	2	S23, S44

Regarding the level of uncertainty (see Table 9), most of the primary studies (i.e., 28 papers) explore uncertainty at the scenario level, 7 papers use only statistical methods to investigate uncertainty, and 12 papers use a combination of both scenarios and statistical methods. Investigating uncertainty at the scenario level is easier to understand, it helps to anticipate potential system behavior in presence of uncertainty, and estimates how the quality requirements may be affected; on the downside it lacks rigorous analysis of system state. Statistical methods, however, can use runtime knowledge to accurately calculate system status in presence of uncertainty, and also enable finding the best adaptation option with the least side effects on quality requirements. Therefore, we envision that using a combination of both scenario and statistical levels will be the most advantageous option for handling multiple quality requirements.

Table 9 – List of papers and level of uncertainty.

Level	Number of papers	Study numbers
Scenario	28	S1, S3, S4, S6, S8, S10, S11, S15, S16, S17, S18, S19, S20, S22, S23, S24, S25, S29, S31, S32, S35, S36, S37, S40, S44, S48, S50, S51
Scenario, Statistical	12	S5, S7, S21, S27, S30, S33, S34, S38, S39, S46, S47, S49
Statistical	7	S9, S12, S13, S14, S26, S42, S43
Not specified	4	S2, S28, S41, S45

Regarding emerging time, Table 10 indicates that most of the existing approaches (i.e., 36 papers) postpone the treatment of uncertainty to the runtime phase. This is not surprising as researchers are mostly interested to study requirements trade-offs at runtime. In 13 papers, uncertainty is treated in both design and runtime. One common way of dealing with uncertainty in these approaches is to acknowledge the existing uncertainty and anticipate probable solutions at design time, but tackle the uncertainty in the runtime phase when more knowledge is available. Finally, we found two papers in which uncertainty is explored and tackled only at design time.

Table 10 - List of papers and the uncertainty treatment time.

Emerging time	Number of papers	Study numbers
Runtime	36	S1, S2, S4, S5, S8, S10, S14, S16, S17, S19, S20, S21, S23, S25, S26, S27, S28, S29, S30, S31, S32, S33, S34, S35, S37, S38, S39, S41, S42, S43, S45, S46, S47, S48, S49, S51
Runtime, Design time	13	S6, S7, S9, S11, S12, S13, S15, S18, S22, S24, S36, S44, S50
Design time	2	S3, S40

Regarding the sources of uncertainty, we note that in some cases there might be an overlap between two or more of the listed sources (e.g., human in the loop, and multiple ownership) definitions; in these cases, we have assigned the primary studies to the most relevant sources. In some cases it is not clear from the paper which source is the most relevant one; in these cases we list the source as hybrid and indicate which multiple sources are applicable. Furthermore listing papers under certain types of sources does not necessarily indicate that the paper provides a solution to tackle uncertainty originating from those particular

sources. It means that the paper discusses uncertainty due to those sources; however, it may or may not propose solutions to resolve uncertainty emerging from one or multiple of those sources.

The most common (i.e., addressed in 38 papers) types of sources of uncertainty in the literature are environmental sources. From Table 11, we see that execution context and human in the loop are respectively the most and the least common sources of uncertainty from the environment uncertainty class. This is not a surprise since the most commonly addressed nature of uncertainty is variability, and variability normally occurs in the execution context of the self-adaptive systems.

Table 11 - List of papers addressing environment uncertainty sources.

Types of environment uncertainty source	Number of papers	Study numbers
Execution context	30	S1, S3, S7, S8, S9, S11, S13, S17, S19, S20, S24, S25, S26, S27, S30, S31, S32, S33, S34, S35, S36, S37, S38, S40, S42, S43, S45, S46, S47, S51
Execution context, Human in the loop	4	S2, S10, S22, S50
Human in the loop	1	S21

Although S6, S18, and S44 address uncertainty originating from environmental sources as well, we could not decide to which source they should be assigned. Therefore, we recoded sources discussed in S6 and S18 as hybrid sources, as they both can be considered uncertainty originating from system and/or environment. Regarding S44, the environmental fact causing the uncertainty is considered as "complexity", despite the rest of the papers which explore the uncertainty due to the dynamicity of the environment.

Table 12 lists sources from goal uncertainty class. Addressed by twelve papers, future goal changes seem to be the most studied goal-related uncertainty in the literature. This suggests that researchers are mainly concerned with the ability of the self-adaptive system to handle its current goals and the potential changes in the future; adding new goals to the system (i.e., future new goals) does not seem to be as important. In Table 12 we also list different sets of sources that we found in the literature; however, the numbers of papers addressing these sets of sources are rather low.

Table 12 - List of papers addressing goal uncertainty sources.

Types of goals uncertainty source	Number of papers	Study numbers
Future goal changes	12	S7, S10, S18, S22, S26, S27, S28, S30, S31, S32, S33, S36
Goal dependency	8	S15, S41, S43, S44, S46, S47, S49, S51
Future new goals	3	S1, S4, S8
Future goal changes, Future new goals	2	S38, S45
Goal dependency, Future new goals	1	S9
Goal dependency, Future goal changes	1	S25
Goal specification, goal dependency	1	S13
Future goal changes, outdated goals	1	S29

S39 and S40 both address the sources achieving stakeholder's objectives and meeting quality of service which can be considered a form of goal uncertainty class. However, we did not assign them to any of our listed sources as it was unclear which sources would be the most relevant. What we noticed from the analysis of goal uncertainty sources is that, although all the included primary studies somehow deal with multiple quality requirements, the trade-off analysis gained little attention in the literature. From 51 selected primary studies, only 8 paper address goal dependencies. In addition, the potential negative impact of self-adaptation on systems quality requirements is not explicitly explored as sources of uncertainty.

Table 13 indicates sources of uncertainty from adaptation functions class. The most commonly discussed (i.e., addressed by 10 papers) source is variability of solution space. This shows that the current focus of research is mainly on providing assurances for applying the best adaptation actions in a system. Self-adaptive systems should be capable of exploring the solution space, and selecting the best solution to adapt the systems with minimum negative side effect on other systems functionalities and quality aspects. Interestingly, the next most common source is fault localization and identification in a system. This suggests that although the most significant source of uncertainty is the selection of most suitable approach for adaptations, in many cases the problem itself, which triggers the need for adaptation, is not identified properly and therefore causes more uncertainty in the system. Sensing and adaptation actions affects are the least common sources from this class of uncertainty. Note that although investigation of adaptation action effects is a major part of resolving the uncertainty due to variability of solution space, and also is a key factor in exploring adaptation effects on quality requirements and handling trade-offs, it has only been explicitly addressed in one paper. These results again confirm the lack of sufficient research on quality requirements trade-off analysis.

Table 13 - List of papers addressing adaptation function uncertainty sources.

Types of adaptation functions uncertainty source	Number of papers	Study numbers
Variability of solution space	10	S9, S13, S15, S25, S26, S39, S42, S43, S44, S48
Fault localization and identification	5	S10, S17, S21, S22, S28
Decentralization	4	S1, S19, S32, S45
Variability of solution space, Fault localization and identification	3	S5, S8, S14
Changes in adaptation mechanisms	2	S31, S35
Variability of solution space, Decentralization	1	S11
Sensing	1	S12
Decentralization, multiple ownership	1	S49
Adaptation action's effects	1	S50

Table 14 presents sources from the model uncertainty class. Our results indicate that uncertainty due to differences in sources of information is the most commonly addressed source in this class. However, we could not find any source which is explored in a significantly higher number of papers; all of the sources from model uncertainty class are discussed in almost equal (low) number of papers.

Table 14 - List of papers addressing model uncertainty sources.

Types of model uncertainty source	Number of papers	Study numbers
Different sources of information	3	S3, S16, S19
Model drift	2	S7, S20
Incompleteness	3	S11, S12, S48
Abstraction	2	S23, S26
Incompleteness, Abstraction	2	S24, S30
Erroneous models	1	S40
Complex models	1	S6

Table 15 presents sources of uncertainty from the resources class. Four papers address changing resources as the origin of uncertainty in self-adaptive system, and one paper deals with newly arrived resources as an uncertainty sources.

Table 15 - List of papers addressing resource uncertainty sources.

Types of resource uncertainty source	Number of papers	Study numbers
Changing resources	4	S2, S21, S22, S28,
New resources	1	S1

Finally, four papers (i.e., S6, S27, S35, and S36) state that sources of uncertainty may be due to systems' circumstances. Complexity in the system is considered as the source of uncertainty in S6, and S27, while systems changes are considered as the sources of uncertainty in S35, and S36.

4. Discussion

In this section, we first present a discussion about sources of uncertainty, and then list the main findings derived from our results and provide implications for researchers.

4.1. Analysis of derived sources of uncertainty based on uncertainty dimensions

One of the major goals of this study was to deliver a comprehensive and well-organized list of commonly addressed sources of uncertainty in self-adaptive systems with multiple quality requirements. Therefore, we believe it is also essential to analyze the derived sources of uncertainty and investigate how each one of these source is handled. In the following, we explore the sources of uncertainty (see Table 5) based on emerging time, level, and nature dimensions. Note that although we performed the same analysis for all classes of sources listed in Table 5, we have omitted results of minor significance.

4.1.1. Environment uncertainty

From 35 papers (see Table 11) that addressed source of uncertainty originating from environment, 10 papers (i.e., S1, S8, S10, S17, S19, S20, S25, S31, S37, and S51) deal with uncertainty at scenario level, due to variability in the context, at runtime. This indicates that variability in the execution context and human behavior are the most common sources of uncertainty, and are mainly handled at runtime. It also shows that researchers mostly use scenarios to understand systems behavior at runtime and resolve the uncertainty. This is an interesting finding as it suggests that statistical methods may be used at runtime to benefit from available knowledge, and study the solutions space to improve the decision making process in self-adaptive systems.

4.1.2. Goals uncertainty

From 29 papers (see Table 12), in which sources of uncertainty originate from goals, eight papers (i.e., S1, S4, S8, S10, S25, S29, S31, and S51) deal with uncertainty due to variability of goals. In these papers, researchers use scenarios to explore how variability may affect the system goals, and deal with the goal uncertainty at runtime. Furthermore, we found that four papers (i.e., S18, S15, S22, and S36) deal with this type of uncertainty both at design and runtime. This indicates that only in a small number of papers (i.e., four papers) researchers manage to touch the issue of goals uncertainty at design time, and in most cases it is postponed to runtime. Despite the fact that more knowledge about system's status is accessible at runtime, statistical solutions are not commonly used to propose rather accurate solutions for handling goals uncertainty at self-adaptive systems with multiple quality requirements. However, a remarkable number of papers (i.e., S7, S27, S30, S33, S38, S46, S47, and S49) use combination of statistical methods and scenarios to deal with goals uncertainty.

4.1.3. Adaptation functions uncertainty

Following the same pattern we found in previous sections, from 28 papers (see Table 12) in which adaptation functions uncertainty sources are addressed, 11 papers (i.e., S1, S8, S10, S17, S19, S25, S31, S32, S35, S48, S49) deal with this class of uncertainty due to variability issues, at scenario level, and at runtime. Interestingly, we found four papers (i.e., S14, S26, S42, and S43) in which statistical methods are used at runtime to deal with adaptation functions uncertainty sources with both variability and epistemic natures. This indicates that although statistical methods are rarely used at runtime, they are favored methods when dealing with adaptation functions uncertainty; specifically, uncertainty due to variability of solution space and fault localization at runtime. Uncertainty due to variability of the solution space is in fact one of the main challenges which needs to be handled when dealing with multiple requirements in self-adaptive systems. The system should be able to manage (i.e., identify, investigate) an increasing number of possible scenarios for adaptation, and predict their effects on quality attributes and select the best adaptation actions. Therefore, it is very crucial to design a self-adaptive system in a way that it collects the most relevant data at a given time and use the right tools to predict the system's behavior in order to handle the quality attributes trade-offs.

4.1.4. Model uncertainty

From 14 papers, in which model uncertainty is addressed, six papers (i.e., S3, S11, S12, S24, S26, and S30) deal with model uncertainty sources due to both variability and lack of knowledge (i.e., epistemic). This is interesting because we have only found 14 papers investigating uncertainty due to combination of both variability and lack of knowledge, and in nearly half of them source of uncertainty is related to models. This shows that lack of knowledge greatly affects credibility of models, and generates uncertainty in self-adaptive systems with multiple quality requirements. From these six papers, three of them (i.e., S11, S12, and S24) deal with uncertainty at both design and runtime, two papers at runtime (i.e., S26, S30), and one paper (i.e., S3) at design time.

It might be interesting for researchers to find methods to use runtime knowledge to constantly adjust and update models. Updated and accurate models are better representatives of the actual self-adaptive systems and ultimately improve the decision making process and trade-off analysis.

4.2. Main Findings and implications for researchers

The following paragraphs elaborate on the main findings, while the end of each paragraph provides implications for researchers in terms of future directions.

Model uncertainty is investigated in both design and runtime. We found that among those approaches which deal with both design time and runtime phases of the system's life cycle, model uncertainty is the most commonly addressed class of uncertainty (see Table 10, and Table 14). From 51 studies, 13 papers (i.e., S6, S7, S9, S11, S12, S13, S15, S18, S22, S24, S36, S44, S50) consider uncertainty in both design and runtime phases, and 5 of these 13 studies (i.e., S6, S7, S11, S13, S24) investigate different types of model uncertainty. This indicates that, although many researchers are focusing on models at runtime to tackle the uncertainty issue, dealing with this particular type of uncertainty (i.e., model uncertainty) is not completely postponed to runtime. In other words, researchers strive to use the available knowledge at design time and probably anticipate system behavior in the future in order to be able to start dealing with model uncertainty

as soon as possible (i.e., design time). Although our results cannot prove the efficiency of this way of combining both design and runtime solutions in dealing with model uncertainty, it confirms its popularity.

Uncertainty is often explored at scenario level regardless of emerging time. Our results show that most of the current studies (i.e., 17 papers) deal with uncertainty at scenario level (see Table 16) at runtime. Researchers frequently try to understand the current state, foresee future behavior of the system, and demonstrate system state during and after application of uncertainty remedy only through scenarios. Surprisingly, approaches expanding through both design time and runtime phases also lack statistical methods. This means that despite the availability of knowledge at runtime, most of these approaches do not consider using statistical methods to reassess their assumptions regarding systems' runtime state in face of uncertainty. Most of the current approaches simply study uncertainty at scenario level (i.e., showcase the behavior of system in the future) through examples, and do not provide rigorous techniques (e.g., probabilistic methods) to support these scenarios. It may be interesting for researchers to further explore incorporating runtime information into statistical methods to mathematically strengthen their anticipations of system behavior.

Table 16 - Emerging time versus level of uncertainty.

	Level	Scenario	Statistical	Both
Emerging Time				
Runtime		17 papers	4 papers	11 papers
Both		9 papers	3 papers	1 paper
Design time		2 papers	None	None

Uncertainty starting to get acknowledged in both design and runtime. Our results indicate that over a decade ago, researchers were focused on solving uncertainty related issues mainly at runtime. This means that both identification of uncertainty and tackling the uncertainty were postponed to the runtime phase. However, around the year 2009 (see Figure 4) researchers started to acknowledge the uncertainty in design time as well. In order to deal with uncertainty in a more structured manner, we propose that researchers investigate whether certain sources of uncertainty can be handled specifically in design or runtime.

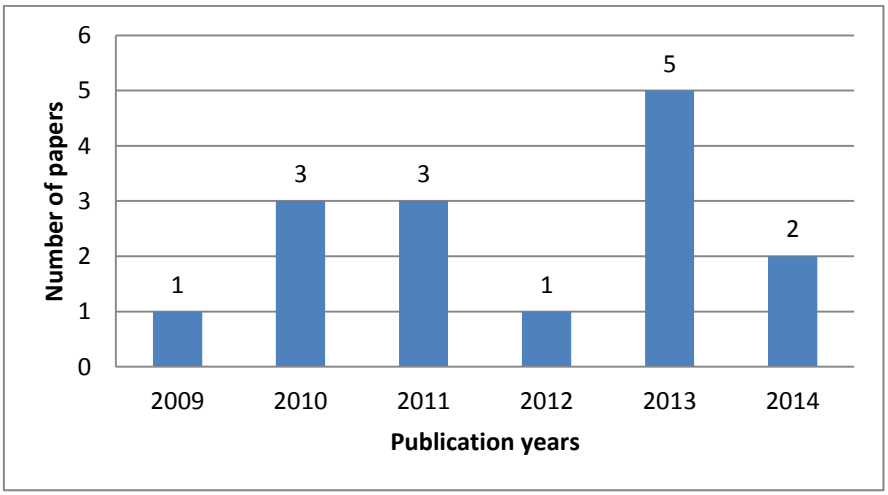

Figure 4 - Number of papers acknowledging uncertainty in design time per year.

Current approaches mainly focus on tackling uncertainty due to variability through approaches in both design and runtime. Variability may emerge in systems requirements, execution environment, or may

be a result of dynamicity of self-systems solutions space. Our results indicate that the main focus of current research is on the variability issues rather than problems originating from lack of knowledge in self-adaptive systems (see Table 8, and Table 10). Therefore, more investigation is required to distinguish the differences in characteristics of variability in different circumstances, and possibly propose tailored solutions capable of dealing better with uncertainty due to variability.

Most commonly addressed source of uncertainty is dynamicity of environment. Not surprisingly, changes in the environment are considered as the main reason behind uncertainty in self-adaptive systems (see Table 11). This is because at the design time, engineers can not anticipate the potential changes in the environment in the future as it is out of their control, and most of the decision making process should be postponed to runtime when more information is available.

Future goal changes is the second most important uncertainty source. From the selected primary studies, we can see that researchers consider changes of system goals as one of the main sources of uncertainty in self-adaptive systems. However, studies rarely explore details of these changes and how changes in one or two goals affects other goals of the system (i.e., requirements trade-offs) explicitly. Therefore, the first step toward handling the requirements trade-offs may be the thorough monitoring of the requirements; this means that adequate data on how the systems' requirements, intentionally or unintentionally, are affected by the adaptation actions (or human's intervene) should constantly be collected, and then the data should properly be analyzed in order to make the best decision and fulfill the requirements at a desired level.

4.3. Limitations of the review and threats to validity

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

4.3.1. Bias

The pilot search indicates that, it is not always easy to extract relevant information from the primary studies. Therefore, there may be some bias and inaccuracy in the extracted data and creation of the classification framework. This is especially prominent for establishing the sources of uncertainty classification due to existing overlap of certain sources definitions. To mitigate this, we included a list of examples from the literature to clarify the sources and help the reader to better comprehend them. Moreover, we had discussions among researchers and asked experts to judge the accuracy of data when the researchers could not reach a consensus on certain extracted data occasionally.

4.3.2. Domain of Study

One of the main risks of performing a systematic literature review 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 to a large extend 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 (Danny Weyns & Ahmad, 2013). To mitigate the risk, we used a generic search string containing all the mostly used terms, and we avoided a much narrowed search string to prevent missing papers in the automatic search. In addition, we established "quasi-gold" standard to investigate the trustworthiness of the created

search string. Furthermore, we also had 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 included them in our final set of selected primary studies.

5. Conclusion and future work

We conducted a systematic literature review to survey the current state of research regarding uncertainty in architecture-based self-adaptive systems with multiple quality requirements. Our results present a classification framework for concept of uncertainty and its different types and categories, and sources of uncertainty in this domain. Furthermore, we investigate the usefulness of the proposed classification framework by analyzing the selected primary studies, and mapping them to the framework. Our work may be interesting for researchers in field of self-adaptive systems as it offers an overview of the existing research and open areas for future work.

Analysis of the selected primary studies suggests that although researchers consider changes of system goals as one of the main sources of uncertainty in architecture-based self-adaptive systems with multiple quality requirements, studies rarely explore details of these changes explicitly and often overlook how changes in one or two of the goals may affect other goals of the system (i.e., requirements trade-offs).

Our results also indicate that uncertainty in architecture-based self-adaptive systems with multiple quality requirements is often explored at scenario level regardless of emerging time of the uncertainty. This means that despite the availability of sufficient knowledge at runtime, most existing approaches do not consider using statistical methods to reassess their assumptions regarding systems' runtime state in face of uncertainty, or incorporate runtime information into statistical methods to mathematically strengthen their anticipations of system behavior in the future. This implies that statistical methods can further be used to more efficiently handle quality requirements and their trade-offs in architecture-based self-adaptive systems tackling uncertainty.

For our future work we plan to particularly focus on uncertainty and its potential influences on quality attributes. To be more specific, we plan to identify types of requirements for which uncertainty in architecture-based self-adaptive systems is more relevant, and investigate the relationship between uncertainty and quality requirements tradeoffs.

Another direction for future work is to focus on proposing methods that are designed to handle a specific class of uncertainty (i.e., uncertainty originating from certain sources) and its sources rather than covering a variety of classes and their sources to a limited degree. Different sources of uncertainty assigned to one class are more likely to overlap, and therefore, focusing on a specific class of uncertainty may result in proposing more structured and efficient methods dealing with multiple sources of uncertainty and their potential interplay.

Appendix

Table 17 - Primary studies included in the review.

Study #	Title	Authors	Year	Venues
1	Architecture-driven self-adaptation and self-management in robotics systems	Edwards G Garcia J Tajalli H Popescu D Medvidovic N Sukhatme G Petrus B	2009	ICSE
2	Self-adaptation for everyday systems	Svein Hallsteinsen, Erlend Stav, and Jacqueline Floch	2004	SIGSOFT
3	Adapt Cases: Extending Use Cases for Adaptive Systems	Luckey M Nagel B Gerth C Engels G	2011	SEAMS
4	A Case Study in Software Adaptation	Valetto G Kaiser G	2002	WOSS
5	Diagnosing architectural run-time failures	Casanova P Garlan D Schmerl B Abreu R	2013	SEAMS
6	Adaptation and Abstract Runtime Models	Thomas Vogel and Holger Giese	2010	SEAMS
7	Dealing with Non-Functional Requirements for Adaptive Systems via Dynamic Software Product-Lines	Ghezzi C Sharifloo A	2013	LNCS
8	A Case Study in Goal-Driven Architectural Adaptation	Heaven W Sykes D Magee J Kramer J	2009	LNCS
9	Designing Search Based Adaptive Systems: A Quantitative Approach	Zoghi P Shtern M Litoiu M	2014	SEAMS
10	Rainbow: architecture-based self-adaptation with reusable infrastructure	David Garlan, Shang-Wen Cheng, An-Cheng Huang, Bradley Schmerl, and Peter Steenkiste	2004	JC
11	Models at Runtime to Support the Iterative and Continuous Design of Autonomic Reasoners	Chauvel, Franck Ferry, Nicolas Morin, Brice	2013	JC
12	Context-aware Reconfiguration of Autonomic Managers in Real-time Control Applications	Anthony R Pelc M Byrski W	2010	ICAC
13	Taming uncertainty in self-adaptive software	Naeem Esfahani, Ehsan Kouroshfar, and Sam Malek	2011	SIGSOFT
14	Architecture-Based Run-Time Fault Diagnosis	Casanova P Schmerl B Garlan D Abreu R	2011	LNCS
15	Requirements and architectural approaches to adaptive software systems: a comparative study	Angelopoulos, Konstantinos Souza, Vítor E. Silva Pimentel, João	2013	SEAMS
16	An architecture for coordinating multiple self-management systems	Garlan D Schmerl B Steenkiste P	2004	WICSA
17	Robust, Secure, Self-Adaptive and Resilient Messaging Middleware for Business Critical Systems	Habtamu Abie, Reijo M. Savola, and Ilesh Dattani	2009	CW
18	A development framework and methodology for self-adapting applications in ubiquitous computing environments	S. Hallsteinsena, , , K. Geihsb, , N. Paspallisc, , F. Eliassend, , G. Horna, J. Lorenzoe, , A. Mamellif, , G.A. Papadopoulosc,	2012	JSS
19	Architecting Self-Aware Software Systems	Faniyi F Lewis P Bahsoon R Yao X	2014	WICSA
20	High-quality specification of self-adaptive software systems	Luckey M Engels G	2013	SEAMS
21	Implementing Adaptive Performance Management in Server Applications	Liu Y Gorton	2007	SEAMS
22	A Framework for Distributed Management of Dynamic Self-adaptation in Heterogeneous Environments	Zouari M Segarra M André F	2010	ICCIT
23	A language for feedback loops in self-adaptive systems: Executable runtime megamodels	Vogel T Giese H	2012	SEAMS
24	Learning revised models for planning in adaptive systems	Sykes D Corapi D Magee J Kramer J Russo A Inoue K	2013	ICSE
25	gocc : A Configuration Compiler for Self-	Nakagawa H	2011	SEAMS

	adaptive Systems Using Goal-oriented Requirements Description			
26	A Learning-based Approach for Engineering Feature-oriented Self-adaptive Software Systems	Elkhodary A	2010	SIGSOFT
27	Towards Run-time Adaptation of Test Cases for Self-adaptive Systems in the Face of Uncertainty	Fredericks E DeVries B Cheng B	2014	SEAMS
28	Model-based Adaptation for Self-Healing Systems	Garlan D Schmerl B	2002	WOSS
29	Improving context-awareness in self-adaptation using the DYNAMICO reference model	Tamura G Villegas N Müller H Duchien L Seinturier L	2013	SEAMS
30	FUSION: a framework for engineering self-tuning self-adaptive software systems	Elkhodary A Esfahani N Malek S	2010	SIGSOFT
31	DYNAMICO: A Reference Model for Governing Control Objectives and Context Relevance in Self-Adaptive Software Systems	Villegas N Tamura G Müller H Duchien L Casallas R	2013	LNCS
32	On decentralized self-adaptation: Lessons from the trenches and challenges for the future	Weyns D Malek S Andersson J	2010	ICSE
33	Improving Architecture-Based Self-Adaptation through Resource Prediction	Cheng S Poladian V Garlan D Schmerl B	2009	LNCS
34	Evolving an adaptive industrial software system to use architecture-based self-adaptation	Camara J Correia P de Lemos R Garlan D Gomes P Schmerl B Ventura R Cámara J	2013	SEAMS
35	Towards Practical Runtime Verification and Validation of Self-Adaptive Software Systems	Tamura G Villegas N Müller H Sousa J Becker B Karsai G Mankovskii S Pezzè M Schäfer W Tahvildari L Wong K	2013	LNCS
36	Model-Driven Engineering of Self-Adaptive Software with EUREMA	Vogel T Giese H	2014	TAAS
37	Achieving dynamic adaptation via management and interpretation of runtime models	Amoui M Derakhshanmanesh M Ebert J Tahvildari L	2012	JSS
38	Towards Self-adaptation for Dependable Service-Oriented Systems	Valeria Cardellini, Emiliano Casalicchio, Vincenzo Grassi, Francesco Lo Presti, Raffaela Mirandola	2009	LNCS
39	Architecture-based self-adaptation in the presence of multiple objectives	Cheng S Garlan D Schmerl B	2006	SEAMS
40	QUAASY: QUality Assurance of Adaptive SYstems	Luckey M Gerth C Soltenborn C Engels G	2011	ICAC
41	Using CVL to Support Self-Adaptation of Fault-Tolerant Service Compositions	Nascimento A Rubira C Castor F	2013	SASO
42	Online Model-based Adaptation for Optimizing Performance and Dependability	Joshi K Hiltunen M Schlichting R Sanders W Agbaria A	2004	SIGSOFT
43	On the relationships between QoS and software adaptability at the architectural level	Perez-Palacin D Mirandola R Merseguer J	2014	JSS
44	Quality attribute tradeoff through adaptive architectures at runtime	Yang J Huang G Zhu W Cui X Mei H	2009	JSS
45	Towards Automated Deployment of Distributed Adaptation Systems	Zouari M Rodriguez I	2013	LNCS
46	A Self-optimizing Run-Time Architecture for Configurable Dependability of Services	Tichy, Matthias Giese, Holger	2004	LNCS
47	Model-Driven Assessment of QoS-Aware Self-Adaptation	Grassi V Mirandola R Randazzo E	2009	LNCS
48	Evaluation of resilience in self-adaptive systems using probabilistic model-checking	Camara J De Lemos R	2012	SEAMS
49	Managing non-functional uncertainty via model-driven adaptivity	Ghezzi C Pinto L Spoletini P Tamburrelli G	2013	ICSE
50	Coupling software architecture and human architecture for collaboration-aware system adaptation	Dorn C Taylor R	2013	ICSE

51	Qos-driven runtime adaptation of service oriented architectures	Valeria Cardellini, Emiliano Casalicchio, Vincenzo Grassi, Francesco Lo Presti, and Raffaela Mirandola	2009	SIGSOFT

Table 18 - List of manually searched venues to create the "quasi-gold" standard.

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

Table 19 - List of automatically searched venues and books.

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)

References

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

Casanova, P., Garlan, D., Schmerl, B., & Abreu, R. (2013). Diagnosing architectural run-time failures, 103–112. Retrieved from http://dl.acm.org/citation.cfm?id=2487336.2487354

Chauvel, F., Ferry, N., & Morin, B. (2013). Models@Runtime to Support the Iterative and Continuous Design of Autonomic Reasoners. In N. Bencomo, R. B. France, S. Götz, & B. Rumpe (Eds.), (Vol. 1079, pp. 26–38). CEUR-WS.org. Retrieved from http://ceur-ws.org/Vol-1079/mrt13_submission_20.pdf

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

Cheung, L., Golubchik, L., Medvidovic, N., & Sukhatme, G. (n.d.). Identifying and Addressing Uncertainty in Architecture-Level Software Reliability Modeling, (MC).

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

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology*, *50*(9-10), 833–859. doi:10.1016/j.infsof.2008.01.006

Edwards, G., Garcia, J., Tajalli, H., Popescu, D., Medvidovic, N., Sukhatme, G., & Petrus, B. (2009). Architecture-driven self-adaptation and self-management in robotics systems. *Software Engineering for Adaptive and Self-Managing Systems, 2009. SEAMS '09. ICSE Workshop on*. doi:10.1109/SEAMS.2009.5069083

Esfahani, N., Kouroshfar, E., & Malek, S. (2011). Taming Uncertainty in Self-adaptive Software. In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering* (pp. 234–244). ACM. doi:10.1145/2025113.2025147

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

Garlan, David. (2010). Software engineering in an uncertain world. *Proceedings of the FSE/SDP workshop on Future of software engineering research - FoSER '10*, 125. doi:10.1145/1882362.1882389

Ghezzi, C, Pinto, L. S., Spoletini, P., & Tamburrelli, G. (2013). Managing non-functional uncertainty via model-driven adaptivity. *Software Engineering (ICSE), 2013 35th International Conference on*. doi:10.1109/ICSE.2013.6606549

Ghezzi, Carlo, & Sharifloo, A. M. (2013). Dealing with Non-Functional Requirements for Adaptive Systems via Dynamic Software Product-Lines. In R. de Lemos, H. Giese, H. A. Müller, & M. Shaw (Eds.), *Software Engineering for Self-*

Adaptive Systems II (pp. 191–213). Springer Berlin Heidelberg. Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-35813-5_8

Hallsteinsen, S., Stav, E., & Floch, J. (2004). Self-adaptation for Everyday Systems. In *Proceedings of the 1st ACM SIGSOFT Workshop on Self-managed Systems* (pp. 69–74). ACM. doi:10.1145/1075405.1075419

Jeffrey, O., & David, M. (2003). The Vision of, (January), 41–50.

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. A., Shaw, M., Andersson, J., Litoiu, M., … Wuttke, J. (2013). Software Engineering for Self-Adaptive Systems: A Second Research Roadmap. In R. de Lemos, H. Giese, H. A. Müller, & M. Shaw (Eds.), *Software Engineering for Self-Adaptive Systems II* (pp. 1–32). Springer Berlin Heidelberg. Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-35813-5_1

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. (2014a). Uncertainties in the Modeling of Self-adaptive Systems : a Taxonomy and an Example of Availability Evaluation, 3–14.

Perez-palacin, D., Milano, P., & Elettronica, D. (2014b). 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.

Tamura, G., Villegas, N. M., Muller, H. A., Duchien, L., & Seinturier, L. (2013). Improving context-awareness in self-adaptation using the DYNAMICO reference model. *Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2013 ICSE Workshop on*. doi:10.1109/SEAMS.2013.6595502

Villegas, N. M., Tamura, G., Müller, H. A., Duchien, L., & Casallas, R. (2013). DYNAMICO: A Reference Model for Governing Control Objectives and Context Relevance 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. 265–293). Springer Berlin Heidelberg. Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-35813-5_11

Vogel, T., & Giese, H. (2010a). Adaptation and abstract runtime models. In *Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems - SEAMS '10* (pp. 39–48). New York, New York, USA: ACM Press. doi:10.1145/1808984.1808989

Vogel, T., & Giese, H. (2010b). Adaptation and Abstract Runtime Models. In *Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems* (pp. 39–48). ACM. doi:10.1145/1808984.1808989

Walsh, W. E., Tesauro, G., Kephart, J. O., & Das, R. (2004). Utility functions in autonomic systems. In *International Conference on Autonomic Computing, 2004. Proceedings.* (pp. 70–77). IEEE. doi:10.1109/ICAC.2004.1301349

Weyns, D, Malek, S., & Andersson, J. (2010). On decentralized self-adaptation: Lessons from the trenches and challenges for the future. In *Proceedings - International Conference on Software Engineering* (pp. 84–93). Dept. of Computer Science,

Katholieke Universiteit Leuven, Belgium. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-77954577834&partnerID=40&md5=1f389e0a603761b96aa46db6bf06e287

Weyns, Danny, & 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, Danny, Malek, S., & Andersson, J. (2012). FORMS: Unifying Reference Model for Formal Specification of Distributed Self-adaptive Systems. *ACM Trans. Auton. Adapt. Syst.*, *7*(1), 8:1–8:61. doi:10.1145/2168260.2168268

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

Zoghi, P., Shtern, M., & Litoiu, M. (2014). Designing Search Based Adaptive Systems: A Quantitative Approach. In *Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems* (pp. 7–16). ACM. doi:10.1145/2593929.2593935