Claims and Evidence for Architecture-Based Self-Adaptation: A Systematic Literature Review

Danny Weyns and Tanvir Ahmad

Department of Computer Science
Linnaeus University, Vaxjo, Sweden
danny.weyns@lnu.se, ta222aw@gmail.com

Abstract. Engineering the upcoming generation of software systems and guaranteeing the required qualities is complex due to the inherent uncertainties at design time, such as new user needs and changing availability of resources. Architecture-based self-adaptation is a promising approach to tackle these challenges. In this approach, a system maintains a model of itself and adapts itself to realize particular quality objectives using a feedback loop. Despite a vast body of work, no systematic study has been performed on the claims associated with architecture-based self-adaptation and the evidence that exists for these claims. As such insight is important for researchers and engineers, we performed a systematic literature review covering 20 leading software engineering conferences and journals in the field, resulting in 121 studies used for data collection. The review shows that self-adaptation is primarily used to improve performance, reliability, and flexibility. The tradeoffs implied by self-adaptation have not received much attention, and evidence is mainly obtained from simple examples. From the study, we derive a number of recommendations for future research in architecture-based self-adaptive systems.

1 Introduction

Engineering the upcoming generation of software systems and guaranteeing the required qualities (performance, robustness, etc.) pose severe challenges due to the inherent uncertainty resulting from incomplete knowledge at design time. Examples of uncertainties are new user needs, subsystems that come and go at will, dynamically changing availability of resources, and faults that are difficult to predict. These challenges have motivated the need for self-adaptive software systems. Self-adaptation endows a system with the capability to adapt itself to internal changes and dynamics in the environment in order to achieve particular quality goals in the face of uncertainty.

Over the past fifteen years, researchers have developed a vast body of work on engineering self-adaptive systems. Two prominent loosely connected approaches to realize self-adaptation are architecture-based self-adaptation and control-based self-adaptation. Architecture-based self-adaptation [14, 11, 9] emphasizes software components for feedback loops, runtime models and mechanisms, and the interaction with the managed system. Control-based self-adaptation [10, 8] applies principles from control theory to design and analyze feedback control loops for computing systems. Our focus in this paper is on architecture-based self-adaptation.

Despite more than a decade of research on self-adaptation, it is not clear how the research results have actually contributed to improvements of engineering complex software systems. Recent efforts resulting from two Dagstuhl seminars summarize achievements in software engineering for self-adaptive systems and outline challenges for future research [6, 13]. But, to the best of our knowledge, no systematic study has been performed on the claims associated with self-adaptation and the evidence that exists for these claims. However, such an insight is crucial for researchers and engineers.

Recently, two related surveys have been conducted. Patikirikorala et al. [15] surveyed engineering approaches for control-based self-adaptation. The authors investigated control methodologies in self-adaptive systems and identified a set of design patterns. However, this survey did not investigate the evidence of self-adaptive systems. Moreover, the survey covered only 9 venues tailored to control-based approaches. In a previous effort [16], we performed a pilot study in which we investigated claimed benefits and supporting evidence for self-adaptation from studies published by the SEAMS community (http://www.self-adaptive.org/) between 2006 and 2012. Most of these studies focus on architecture-based self-adaptation. While this pilot provided useful insights for the SEAMS community, the survey was limited in scope and time and as such did not provide conclusive insights for the field as a whole.

The goal of the research presented in this paper is to perform a comprehensive study, aiming to identify:

1. The focus of research on architecture-based self-adaptation,
2. The claimed benefits of architecture-based self-adaptation,
3. The evidence that is provided for these claims.

To that end, we have performed a systematic literature review. In this review we searched 20 main software engineering venues and journals in the period 2000-2012, resulting in 121 primary studies for data collection. All material of the systematic literature review is available at the survey website.[1]

Paper Overview. Section 2 provides a short introduction of architecture-based self-adaptation. In Section 3, we describe the method we used in our research. In Section 4 we present and analyze the data extracted from the primary studies to answer the research questions. Section 5 discusses limitations of our study. Finally, we derive conclusions from the review and highlight a number of recommendations for future research in architecture-based self-adaptation in Section 6.

2 A Brief Introduction to Architecture-based Self-Adaptation

Figure 1 shows the primary elements of a self-adaptive system situated in an environment. We use the general terms *managed* subsystem and *managing* subsystem to denote the constituent parts of a self-adaptive software system [11, 9, 17].

The environment refers to the part of the external world with which the self-adaptive system interacts and in which the effects of the system will be observed and evaluated.

[1] http://homepage.lnu.se/staff/daweaa/SLR/CESAS/CE-SAS.htm

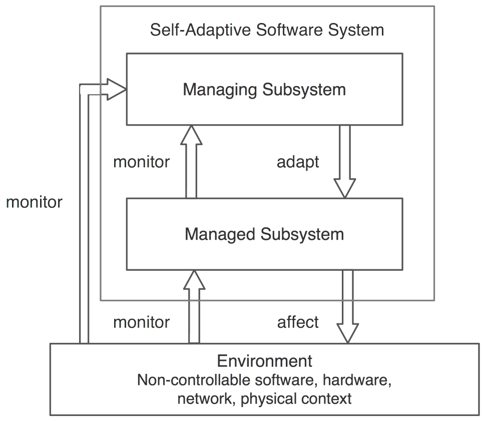
Self-Adaptive Software System
Managing Subsystem
monitor
adapt
monitor
Managed Subsystem
monitor
affect
Environment
Non-controllable software, hardware, network, physical context

Fig. 1. Constituent parts of a self-adaptive software system.

The distinction between the environment and the self-adaptive system is made based on the extent of control. The managed subsystem comprises the application logic that provides the system's domain functionality. The managing subsystem manages the managed subsystem. The managing subsystem comprises the adaptation logic that deals with one or more concerns. To realize its goals, the managing subsystem monitors the environment and the managed subsystem and adapts the latter when necessary. Other layers can be added to the system where higher-level managing subsystems manage underlying subsystems, which can be managing subsystems themselves. One common approach to describe the functions of managing subsystems is by means of a Monitor-Analyze-Plan-Execute-Knowledge loop [11] (MAPE-K loop). The MAPE elements map to the basic functions of a feedback loop, while the K component maps to runtime models maintained by the managing system to support the MAPE functions [17].

It is important to note that the managed and managing subsystems can be deployed centralized or distributed, and both subsystems can be explicitly separated or they can be (partially) interwoven. Furthermore, the managing system can consist of one or more feedback loops, and the MAPE functions can be mapped to distinct components, or they can be integrated in one or more components.

3 Research Method

Our study uses a systematic literature review [12], which is a well-defined method to identify, evaluation and interpreting all relevant studies regarding a particular research question or topic of interest. A systematic literature review comprises three main phases: planning, executing, and reporting. In the planning phase, the protocol for the review is defined. This protocol describes the procedure that will be followed to conduct the review. In the execution phase, studies are selected, data is extracted, and the results are analyzed. In the reporting phase, the study results are documented.

Three researchers were involved in the systematic literature review. The team defined the protocol. To minimize bias, each primary study was assigned to two researchers that independently collected the data. During discussion sessions with the three reviewers, the collected data was compared and in case of differences, conflicts were resolved. The data was then entered in a data base system for further processing. Data analysis was performed by two researchers and discussed with the third researcher. Finally, two researchers produced the final report of the review. The report was checked by the third researcher and adjustments were made where needed.

We now discuss the research questions, searched sources, search strategy, inclusion and exclusion criteria, collected data items, and approach for data analysis.

3.1 Research Questions

We formulated the goal of the review using the Goal-Question-Metric (GQM) perspectives (purpose, issue, object, viewpoint) [3]:

Purpose: Analyze and characterize
Issue: the claims and evidence
Object: for architecture-based self-adaptive software systems
Viewpoint: from a researcher's viewpoint.

This overall goal can be translated to three concrete research questions:

RQ1: What is the focus of research in architecture-based self-adaptation?
RQ2: What are the claims made for self-adaptation and what are the tradeoffs implied by self-adaptation?
RQ3: How much evidence is available for the claims and what are the types of evidence?

With RQ1, we want to get insight in the trends of research on architecture-based self-adaptation and the current state of the art. RQ2 is motivated by the need to get clear understanding of the benefits of architecture-based self adaptation, that is, we are interested in identifying which concerns are addressed in self adaptive systems and what are the tradeoffs implied by applying self-adaptation. With RQ3 we aim to investigate what assessment methods have been used to obtain evidence for the research results and how much evidence is available for the applied methods.

3.2 Searched Sources

To guarantee high quality of the primary studies and obtain solid data to answer the research questions, we searched the main conferences and journals for publishing research results on self-adaptive systems, software architecture, and software engineering. The selected sources are listed in Table 1. Rank is based on the Australian Research Council ranking and H-index[2]. Instead of a general search, we opted for searching the main specialized venues and the premier software architecture and engineering venues, guaranteeing inclusion of high-quality primary studies for data collection.

[2] ARC: http://www.arc.gov.au/era/era_2010/archive/era_journal_list.htm,
H-index: http://www.scimagojr.com and http://academic.research.microsoft.com/

Table 1. Searched Sources

ID	Conference/Journal	Rank	H-index
Adaptive	Adaptive and Self-adaptive Systems and Applications	n/a	n/a
ASE	International Conf. on Automated Software Engineering	A	24
DEAS	Design and Evolution of Autonomic Application Software	n/a	n/a
ECSA	European Conference on Software Architecture	n/a	8
FSE	Foundations of Software Engineering	A	31
ICAC	International Conference on Autonomic Computing	B	32
ICSE	International Conference on Software Engineering	A	63
ICSM	International Conference on Software Maintenance	A	57
ISARCS	International Symposium on Architecting Critical Systems	n/a	n/a
ISSTA	International Symposium on Software Testing and Analysis	A	35
SASO	Self-Adaptive and Self-Organizing Systems	n/a	9
SEAMS	Software Engineering for Adaptive & Self-Managing Systems	n/a	n/a
SefSAS	Software Engineering for Self-Adaptive Systems	n/a	n/a
WADS	Workshop on Architecting Dependable Systems	n/a	n/a
WICSA	Working International Conference on Software Architecture	A	n/a
WOSS	Workshop on Self-Healing	n/a	n/a
JSS	Journal of Systems and Software	A	48
TAAS	Transactions on Autonomous and Adaptive Systems	n/a	16
TOSEM	Transactions on Software Engineering and Methodology	A*	47
TSE	Transactions on Software Engineering	A*	93

3.3 Search Strategy

The search strategy combines automatic with manual search. In a first step we searched primary studies by automatic search using the following search string:

> ((Title:adaptive OR Title:adaptation OR Title:self OR Title:autonomic OR Title:autonomous) OR
> (Abstract:adaptive OR Abstract:adaptation OR Abstract:self OR Abstract:autonomic OR Abstract:autonomous))

We performed automated search on three data search engines: IEEE Explore, ACM Digital Library, and Springer for the respective venues. Search was based on title and abstract. To ensure that the search string provides the right scope of studies, we applied pilot searches on the set of studies from three venues: TAAS, ICAC, and SEAMS.

In the second step, two researchers read the abstracts, introduction and conclusions of all the primary studies selected in the first step and used the inclusion/exclusion criteria to filter out the studies that were not relevant for the review. For a number of papers, we further looked into other sections. We explain the selection criteria below.

3.4 Inclusion and Exclusion Criteria

We used the following inclusion criteria in our search:

- Studies which were published between January 2000 to December 2012. We used 2000 as starting date as self-adaptive systems have become subject of active research around that time.
- Studies on self-adaptive systems that at least partially separate the managing system (adaptation logic) from the managed system (domain logic).
- Studies that concern the engineering of self-adaptation, i.e. the realization of self-adaptation or parts of self-adaption.
- Studies that provide a minimal level of assessment of the research, which may be in the form of example application, simulation, rigorous analysis, empirical, or real world example.

We used the following exclusion criteria:

- Surveys and roadmap papers, as we are only interested in studies that provide a minimal level of assessment of research results.
- We also excluded tutorials, short papers, editorials etc. because these papers do not provide reasonable data.

A paper was selected as a primary study if it met all inclusion criteria and eliminated if it met any exclusion criterion.

3.5 Data Items

Table 2 shows the data items we extracted to answer the research questions. For each research question, we identified 3 to 4 data items that aim to provide data to answer the research question. Several of these data items are defined based on the insights derived from the pilot study [16].

We briefly discuss the different data items. The concrete options for each data item are further discussed in the next section. For a detailed description of the data items, we the protocol that is available at the survey website.

F1-F5: The data items author(s), year, title, venue, citation count are used for documentation.

F6: Quality score assesses the quality of study, which is important for data analysis and interpretation of results. Based on [7] and the pilot study, we assessed the following quality items: (1) problem definition of the study, (2) problem context, i.e., the way the study is related to other work, (3) research design, i.e., the way the study was organized, (4) contributions and study results, (5) insights derived from the study, (6) limitations of the study. For each item, we have quality levels: explicit description (2 points), general description (1 point), and no description (0 points). A quality assessment score (max 12) is calculated by summing up the scores for all the items for a study.

Table 2. Data Items

Item ID	Field	Use
F1	Author(s)	Documentation
F2	Year	Documentation
F3	Title	Documentation
F4	Venue	Documentation
F5	Citation count	Documentation
F6	Quality score	RQ1-3
F7	Subject of the study	RQ1
F8	Feedback loop architecture	RQ1
F9	Application domain (if applicable)	RQ1
F10	Quality concerns	RQ2
F11	Claimed benefits	RQ2
F12	Tradeoffs	RQ2
F13	Validation setting	RQ3
F14	Assessment approach	RQ3
F15	Evidence level	RQ3
F16	Repeatability	RQ3

F7: Subject of the study refers to the software engineering field that is addressed in the study. We used the SWEBOK sub-disciplines [1] to define the options, including software requirements, software design, software construction, software testing, software maintenance, among others.

F8: Feedback loop architecture refers to the structure of the feedback loop(s) (or parts of it) that are the focus of the study. Options range from: focus on particular MAPE functions, to single MAPE loop, and mutiple MAPE loops.

F9: Application domain refers to the kind of application for which self-adaptation is used. We started from the an initial list of application domains taken from our pilot study [16] and added additional domains when they appeared during the review.

F10: Quality concerns refer to the concerns related to self-adaptation. We defined the following option based on IEEE 9126 and ISO/IEC 25012: reliability, availability, usability, efficiency/performance, maintainability, portability, security, accuracy, flexibility, and other concern.

F11: Claimed benefits refer to the concerns of self-adaptation (identified in F10) with positive impact. Options are: preserving quality of the software, improving quality of the software, assuring quality of the software, and improving other concerns.

F12: Tradeoffs refer to the concerns of self-adaptation (identified in F10) with a negative impact. Option are: quality concerns that are negatively influenced due to self-adaptation, and other concerns that are negatively influenced due to self-adaptation.

F13: Validation setting refers to the context in which validation is performed, with the options: academic effort, academic/industry collaboration, and industrial effort.

F14: Assessment approach refers to the method used for evaluating the research results. Options are: example application, simulation (use of a model of the real world), rigorous analysis (typically based on formal methods), empirical study (case study, controlled experiment), and experience from real examples.

F15: Evidence level expresses the degree of evidence for the research results. Evidence can be obtained from: demonstration or application to simple examples, expert opinions or observations, empirical studies, and industrial evidence.

F16: Repeatability of the study is one of the following options: the study is not repeatable (no useful material is available to repeat the study), a partial description is available to repeat the study, the material to repeat the study is partially available, all the material is available to repeat the study.

3.6 Approach for Analysis

The data items of the primary studies was collated to answer the research questions. Analysis included: (i) obtaining consensus among the reviewers in case of conflicts, (ii) analyzing the data, for which we used descriptive analysis and multiple regression to identify correlations, and (iii) answering research questions. Based on the analysis results, we derived conclusions and recommendations for future research in the area of architecture-based self-adaptation, and we reflected on threats to validity of the review.

4 Results Analysis

We start by giving an overview of the primary studies selected for the review. Then we discuss the results for each research question.

4.1 Selected Primary Studies

From 7400 studies published at 20 conferences/journals we retrieved 1296 studies after applying the search string. From these studies we selected 121 primary studies after applying the inclusion/exclusion criteria. A list with the selected primary studies is available at the survey website. Figure 2 shows the number of selected studies per venue.

We see that JSS is the most popular journal to publish papers on architecture-based self adaptive systems with 21.5% of the studies, while SEAMS is the most prominent conference with 19.9% of the studies. TSE and TAAS represent 14.9% of the studies and the top software engineering conferences ICSE, FSE and ASE represent 9.9% of the studies. The architecture focused venues, WICSA, ECSA, and ISARCS represent 6.7% of the studies. 10.7% of the studies were published between 2000 and 2005 and 89.3% between 2006 and 2012, which shows the growing research interest in this area.

Figure 3 summarizes the quality scores for the selected primary studies. The results show that researchers provide descriptions of the problem they tackle and how the problem relates to other efforts. Contributions and insights are also reported, although not

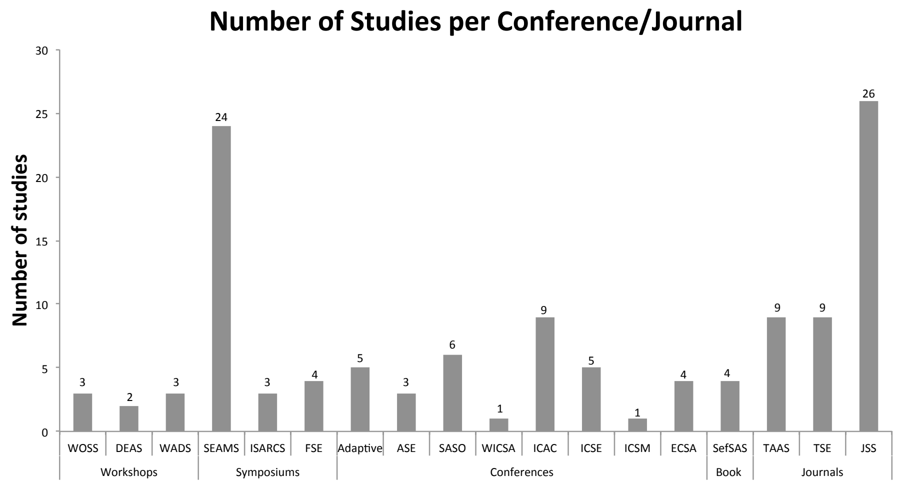

Fig. 2. Primary studies per venue/journal.

always explicitly. However, the majority of studies do not describe research design, i.e. the way the research is organized, and most studies ignore reporting limitations of the results (although we notice that a growing number of researchers have started reporting limitations after 2008). Providing an explicit description of research design is common practice for empirical studies, but less common in software engineering in general. The results confirm this trend for the primary studies in this review. However, the poor treatment of limitations deserves attention as this should be a key part of any engineering study. Table 3 shows the regression analysis between the number of studies and quality score for different publication fora.

Table 3. Number of studies and quality score for different publication fora

Venues	Regression Eq.	R	Mean	S.D.
Journals	y = -0,0672x + 2,3498	-0,11	6,11	1,7
Conferences	y = -0,1502x + 2,4545	-0,27	5,39	1,68
Symposia	y = -0,1067x + 1,998	-0,15	5,67	1,37
Book Chapters	y = -0,0178x + 0,2895	-0,16	5,38	1,75
Workshops	y = -0,0791x + 0,9051	-0,37	4,28	1,33

The values confirm common sense that the primary studies with the best quality scores are published in journals, while studies presented at workshops have lower quality scores. However, with a mean of the overall score of 5.6 (on a max of 12), the quality of the selected primary studies can be considered as reasonably good.

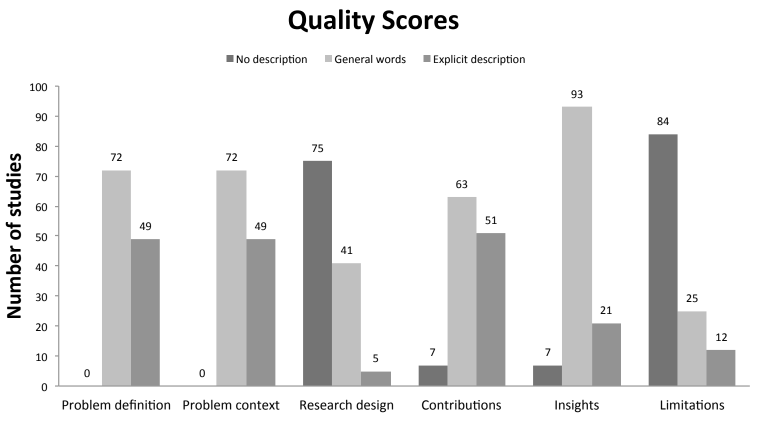

Fig. 3. Quality scores for the primary studies

4.2 RQ1: What is the focus of research in self-adaptation?

Research focus is derived from data items: subject of the studies (F7), feedback loop architecture (F8), and application domain (F9).

The most popular subject of the studies (F7) in terms of SWEBOK software engineering fields is software design with 48% of the studies, followed by software quality with 17%, software requirements with 8% and software testing with 8%. Design activities are an evident focus of architecture-based self-adaptation. Requirements for self-adaptive systems have gained increasing attention during the last years (all studies on requirements are from 2006 onward), confirming that handling dynamic changing user needs is a topic of increasing importance in software engineering.

Figure 4 shows the frequency of feedback loop architecture (F8). The dominant focus has been on single feedback loops, with 37% of the studies using distinct components for each of the MAPE functions and 32% using components that mix (some of) the MAPE functions. 20% of the studies (24 in total) focus on multiple feedback loops. All studies directly or indirectly refer to the MAPE functions in their solutions, which shows that MAPE serves a *reference model* (i.e., a division of functionality together with flows between the pieces [4]). However, as a significant number of studies do not map these functions one-to-one to components, MAPE is not generally considered as a *reference architecture* (i.e., a reference model mapped to software elements). The numbers show that researchers have payed less attention to engineering self-adaptive systems with multiple control loops. However, we notice that 92% of these studies have been published in the last four years, which underpins the growing interest in this area.

Figure 5 shows the frequency of application domains (F9). Only 69% of the studies do consider an explicit application domain. The remaining studies refer to abstract applications, such as resource management, service-based system, networking, etc. The dominant application domains are embedded systems (46%) and web applications (30%); the latter are e-commerce (such as travel planning, book store, etc.) and information systems (such as news services, social media, etc.). Embedded systems

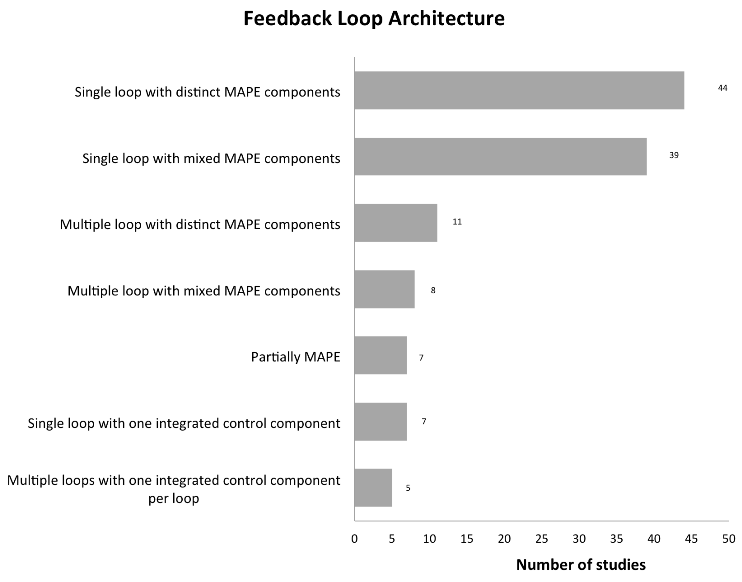

Fig. 4. Feedback loop architectures

have always been an important domain in research on self-adaptation. In the last years, dynamic service composition has gained increasing attention. We found that 86% of the studies with multiple feedback loops are applied to the domains of embedded systems, traffic, and robotics, which can be explained by the fact that these domains are characterized by loosely coupled, physically distributed entities.

Summary for RQ1: The main focus of research in engineering architecture-based self-adaptation has been on software design of a single feedback loop, applied to the domains of embedded systems and web applications. Driven by the engineering challenges of future software systems, there is a growing interest in requirements for self-adaptive systems, dynamic service composition, and multiple feedback loops.

4.3 RQ2: What are the claims made for self-adaptation and what are the tradeoffs implied by self-adaptation?

The answer to RQ2 is derived from quality concerns (F10), claimed benefits (F11), and tradeoffs (F12).

The top three concerns related to self-adaption (F10) are efficiency/performance (55% of the studies), reliability (41%), and flexibility (28%). Accuracy, security, usability, maintainability, and availability account each for 6% or less of the studies. Other reported concerns are engineering effort, complexity, stability, and cost. These concerns are considered in only 6% of the studies (in total). This latter observation is remarkable as seminal papers in the area of self-adaptation use these other concerns as the primary arguments for the need of self-adaptation [14, 11, 9].

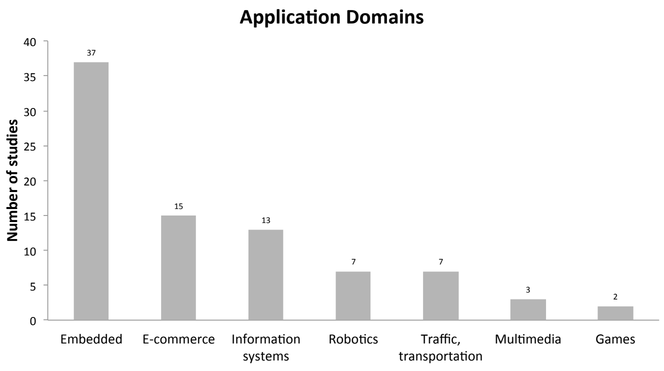

Fig. 5. Studied application domains

We analyzed the correlation between the main quality concerns and the main application domains. Table 4 shows the results of this regression analysis.

Table 4. Correlation between main quality concerns and application domains

Application Domains	Efficiency/Performance	Reliability	Flexibility
Embedded	0,89	0,84	0,59
Information Systems	0,88	0,68	0,78
E-commerce	0,75	0,63	0,81

The results tell us that efficiency/performance is relevant to self-adaptation in all primary domains, while reliability is more relevant to embedded systems and flexibility to web-based systems. Reliability is a classic quality concern in embedded systems. On the other hand, in web-based systems, flexibility provides an alternative for reliability tailored to open environments. For example, a common approach to deal with uncertainty about the availability of services is to exploit self-adaptation to replace dynamically a service that becomes unavailable.

We also looked at the number of concerns considered in individual studies and measured that 57% of the studies consider a single concern, 40% consider 2 concerns, the remaining 3% consider more concerns. We can conclude that most researchers take a narrow view on engineering self-adaptive systems, focusing on a particular concern, without considering the interplay with other concerns.

Figure 6 summarizes the data for claimed benefits (F11) and tradeoffs (F12). This important figure clearly shows that most studies focus on concerns with a positive effect, i.e., 91% of the concerns related to self-adaptation are claimed to be positively influenced. Broken down, 81% of the studies state that a quality of the software is improved by self-adaptation, 5% state that a quality is assured, and the remaining 5% state that a quality is preserved.

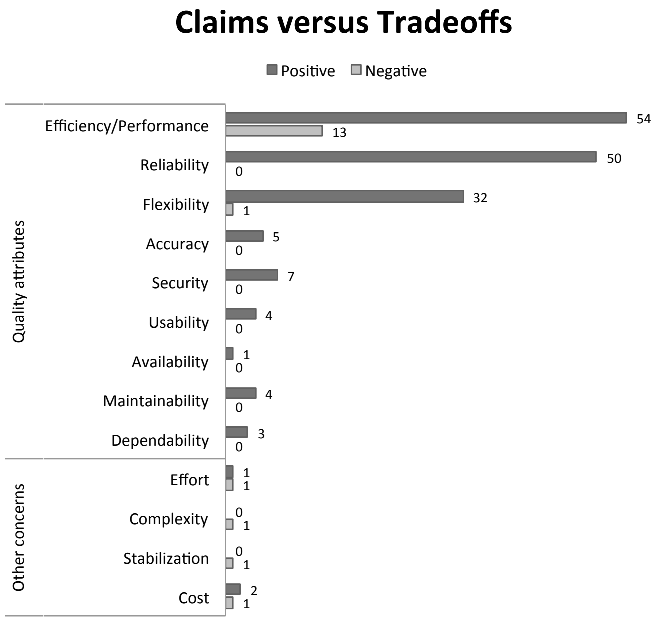

Fig. 6. Claims and tradeoffs of self-adaptation

On the other hand, little attention is given to concerns with a negative effect, i.e., the tradeoffs implied by self-adaptation. 10.7% of the studies state that self-adaptation has an efficiency/performance cost, a single study considers a negative effect on flexibility, and 3.3% of the studies state a negative effect on other engineering aspects (effort, complexity, stability and cost). Concretely, seven studies report an efficiency/performance tradeoff against flexibility and six studies against reliability. Three the four studies report negative effects to other concerns against performance, the other one against accuracy.

This analysis confirms that the majority of researchers focus on a single concern only (see F10). Even if multiple concerns are considered, they mainly look at the positive effects of self-adaptation. To further understand these observations, we looked into the studies and found that 80% of the studies that do not consider tradeoffs in their evaluation, simply ignore implications of self-adaptation. 13% of the studies recognize possible tradeoffs and acknowledge the limitations of their study in that respect, the other 7% of the studies postpone the issues related to tradeoffs to future work.

Summary for RQ2: Most researchers on self-adaptive systems claim improvements of software qualities, in particular for efficiency/performance, reliability, and flexibility. Tradeoffs are hardly considered at all, neither with respect to other qualities nor the effects on concerns such as effort and cost. A minority of researchers recognize the limitations of their work with respect to tradeoffs or they postpone it to future work.

4.4 RQ3: How much evidence is available for the claims and what are the types of evidence?

To answer this question, we analyze the data extracted from validation setting (F13), assessment approach (F14), evidence level (F15) and repeatability (F16).

For validation setting (F13), we found that out of 121 studies, only two studies were performed in a joint effort between academic and industry. No industry-only studies have been reported. These numbers give a strong indication that the research results of architecture-based self-adaptation have not found their way to practice (at least, they have not been reported in the main software engineering venues).

Figure 7 shows the assessment methods that have been used in the studies (F14). Example application accounts for 67.8% of the studies, simulation for 19.8%, rigorous analysis for 8.3%, empirical study for 2.5% and experience from real-world example for 1.7%. Closer examination reveals that almost all studies use simple basic example applications to assess the research findings. The reported empirical studies were in fact quasi empirical studies. No controlled experiments have been reported in the area of engineering architecture-based self-adaptation and experiences with real-world examples is very limited. The lack of both empirical evidence and studies with industry partners hampers industrial adoption of architecture-based self-adaptation in general.

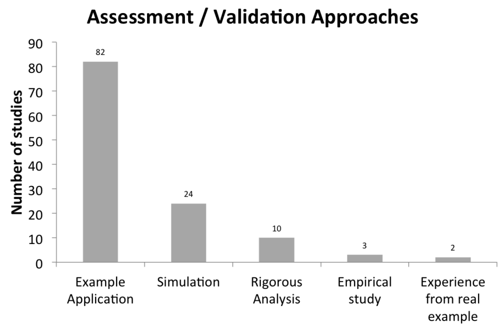

Fig. 7. Assessment approaches

Table 5 shows that example applications are used in all application domains. Simulation is mainly used in web-based systems (e-commerce and information systems), while rigorous analysis is mainly used for embedded systems and e-commerce.

Given the used assessment methods, it is not surprising that most studies have a low evidence level (F15). Concretely, 95.8% of the studies provide minimal evidence from demonstrations or simple/toy examples, 1.7% provide evidence from expert opinions or observation, and 2.5% provide (weak) empirical evidence.

Summary for RQ3: Most research on architecture-based self-adaptive systems is assessed using simple example applications with a minimal level of evidence. Few empirical studies exist and there is hardly any industrial application of architecture-based self-adaptation reported. Weak evidence and poor connection with practice shows that research in architecture-based self-adaptation is still more exploratory than exploitative.

Table 5. Correlation between assessment methods and application domains

Assessment Methods	Embedded	Robotics	E-commerce	Traffic and transport.	Information systems
Example Application	0,93	0,93	0,84	0,88	0,85
Simulation	0,78	0	0,84	0,14	0,82
Rigorous Analysis	0,73	0	0,86	0	0,20

5 Limitations of Study

Despite the sound methodology, this study has some limitations. First, our study is limited to 20 major venues in the field. While we believe that these are the most prominent venues for research on architecture-based self-adaptive systems, we may have missed a number of primary studies that have been published elsewhere. Second, we used common terms to formulate the search string. However, these terms may not fully cover all studies on architecture-based self-adaptation, as there is no generally agreed consensus on the key terms in the field. This limitation is inherent to a field where research is still in an exploratory phase. To minimize this threat, we performed a number of pilot searches to get optimal coverage of automatic search. Third, there is a potential bias of the reviewers. We believe that the comprehensive selection and data extraction process that involved two reviewers who cross-checked the search results, supported by a third reviewer to obtain consensus in case of conflicts, should minimize this threat of bias.

6 Conclusion

Research on architecture-based self-adaptation is widely recognized as key for tackling several of the hard challenges we currently face in software engineering. However, reflecting on the results and analysis of our study, we conclude that there are opportunities for improving coherence in research to move the field forward. We recommend coherence improvements in three dimensions.

First, coherence among the researchers can be improved. We observe that different groups follow specific lines of research that are only weakly connected. Researchers apply their results to specific applications and mostly ignore limitations. Furthermore, there is a lack of empirical studies. Clear and fair treatment of limitations and evidence for findings provide a basis for both consolidation of results and starting points for future research efforts in the field. However, there are some positive signs. First, we notice that researchers have started reporting limitations of their work. Over 85% of the studies that report limitations have been published since 2008. Furthermore, a recent study [18] reports the results of a first controlled experiment on design improvements of using external feedback loops to realize architecture-based self-adaptation.

Second, coherence of research that spans software engineering fields can be improved. We observe a clear dominance of attention for the design of self-adaptive systems. Clearly, there is a need to integrate design with other engineering activities of self-adaptive systems, including requirements, testing and engineering processes. Here too, we observe some positive signs. During the last years, we notice a growing interest in the study of requirements for self-adaptive systems, lead by different groups in the world. We also notice a growing interest in other activities, e.g. the 10 studies on testing were all published since 2008. Finally, a recent publication [2] shows an interest of the community in engineering processes for self-adaptive systems.

Third, coherence of research with the surrounding world can be improved. Currently, research is primarily evaluated using simple applications without making the material available to others. Worse, collaborations with industry partners are very rare. Availability of experimental material and industrial involvement are essential to the field to obtain maturity. But again, there is some hope. The community took the initiative to establish *exemplars* that provide model problems for the community (http://seams.self-adapt.org/wiki/Exemplars). We also refer to a recent study [5] that reports experiences of an industrial application of architecture-based self-adaptation.

We performed a systematic literature review study that shed light on the claims that are made for architecture-based self-adaptation and evidence that is provided for these claims. We hope that this study can contribute to push this important field forward.

References

1. Abran, A., et al. (eds.): Guide to the Software Engineering Body of Knowledge - SWEBOK. IEEE Press, Piscataway, NJ, USA (2001)
2. Andersson, J., et al.: Software eng. processes for self-adaptive systems. LNCS 7475 (2013)
3. Basili, V., et al.: Goal question metric approach. In: Encyclopedia of Soft. Eng. (1994)
4. Bass, L., et al.: Software Architecture in Practice. Addison Wesley (2003)
5. Camara, J., et al.: Evolving an adaptive industrial software system to use architecture-based self-adaptation. SEAMS (2013)
6. Cheng, B.H., et al.: Software engineering for self-adaptive systems: A research roadmap. LNCS vol. 5525, Springer (2009)
7. Dybå, T., Dingsøyr, T.: Empirical studies of agile software development: A systematic review. Inf. Software Technology 50, 833–859 (2008)
8. Filieri, A., et al.: Self-adaptive software meets control theory: A preliminary approach supporting reliability requirements. In: ASE (2011)
9. Garlan, D., et al.: Rainbow: Architecture-based self-adaptation with reusable infrastructure. IEEE Computer 37, 46–54 (2004)
10. Hellerstein, J., et al.: Feedback Control of Computing Systems. Wiley (2004)
11. Kephart, J., Chess, D.: The vision of autonomic computing. Computer 36(1) (2003)
12. Kitchenham, B., Charters, S.: Guidelines for performing systematic literature reviews in software engineering (EBSE 2007-001, Keele and Durham University) (2007)
13. de Lemos, R., et al.: Software engineering for self-adaptive systems: A second research roadmap. LNCS vol, 7475, Springer (2012)
14. Oreizy, P., et al.: Architecture-based runtime software evolution. In: ICSE (1998)
15. Patikirikorala, T., et al.: Survey on the design of self-adaptive software systems using control engineering approaches. SEAMS (2012)
16. Weyns, D., et al.: Claims and supporting evidence for self-adaptive systems: A literature study. Software Engineering for Adaptive and Self-Managing Systems (2012)
17. Weyns, D., et al.: Forms: Unifying reference model for formal specification of distributed self-adaptive systems. ACM TAAS (2012)
18. Weyns, D., et al.: Do external feedback loops improve the design of self-adaptive systems? a controlled experiment. In: SEAMS (2013)