How do we Evaluate Self-adaptive Software Systems?

A Ten-Year Perspective of SEAMS

Ilias Gerostathopoulos[1], Thomas Vogel[2], Danny Weyns[3], and Patricia Lago[1]

[1]*Vrije Universiteit Amsterdam, Netherlands,* [2]*Humboldt-Universität zu Berlin, Germany,* [3]*KU Leuven, Belgium*

Abstract—**With the increase of research in self-adaptive systems, there is a need to better understand the way research contributions are evaluated. Such insights will support researchers to better compare new findings when developing new knowledge for the community. However, so far there is no clear overview of how evaluations are performed in self-adaptive systems. To address this gap, we conduct a mapping study. The study focuses on experimental evaluations published in the last decade at the prime venue of research in software engineering for self-adaptive systems—the International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS). Results point out that specifics of self-adaptive systems require special attention in the experimental process, including the distinction of the managing system (*i.e.,* the target of evaluation) and the managed system, the presence of uncertainties that affect the system behavior and hence need to be taken into account in data analysis, and the potential of managed systems to be reused across experiments, beyond replications. To conclude, we offer a set of suggestions derived from our study that can be used as input to enhance future experiments in self-adaptive systems.**

I. Introduction

Increasingly, we expect software-intensive systems be able to change their structure and behavior at runtime to continue meeting their goals while operating under uncertainty—they need to be self-adaptive. Self-adaptation is typically realized via feedback loops that continuously monitor a system and enact changes to the system. Self-adaptation has been an active area of research for over 20 years [74], initiated by IBM's pioneering vision of autonomic computing [34] and the seminal work of Oreizy *et al.* [46] and Garlan *et al.* [24].

Numerous new approaches focusing on a variety of aspects of engineering self-adaptive systems (runtime models [59], modeling languages [70], verification at runtime [8], planning [47], etc.) have been proposed by the research community. To that end, a set of exemplars and reusable artifacts were developed for use by the self-adaptive systems community.[1] Given this substantial body of work in the area, it is important to obtain a clear view of how contributions have been evaluated. While related work has shed light on some aspects of evaluation, *e.g.,* [6], [50], [56], to the best of our knowledge, no study has targeted an in-depth analysis and characterization of the way experimental evaluations have been conducted.

Evaluation is central to self-adaptive systems (as for any type of software systems), since novel approaches must be assessed based on their contribution [5]. Yet, evaluating contributions of self-adaptive systems may raise particular challenges due to the specifics of these systems [13] (*e.g.,* the use of feedback loops to realize adaptation) and their ability to deal

with uncertainty during operation [7]. Understanding the state of the art in conducting evaluations in self-adaptive systems enables researchers to better compare new findings. Hence, it is important to provide a systematic overview of evaluations of self-adaptive systems, which is currently missing.

To fill this gap, we performed a mapping study [51] aimed to address the question "How do we evaluate self-adaptive software systems?" We focus on experimental evaluations, *i.e.,* evaluations that use one or more experiments, since experiments are the most common evaluation approach used in self-adaptive systems. Concretely, the study is centered on (i) the scope of experiments, (ii) the way experiments are designed and operated, and (iii) the way the results of such experiments are analyzed, and (iv) packaged.

The remainder of this paper is structured as follows. Section II presents background and related work. In Section III, we summarize the study protocol, including research questions and process. Section IV presents the results of the study and answers the research questions. In Section V, we discuss insights and threats to validity, and we conclude in Section VI.

II. Background and Related Work

A. Basic Concepts of Self-Adaptive Systems and Experiments

This study focuses on what is known as architecture-based adaptation [24], [37], [46], *i.e.,* a widely applied approach to realize self-adaptation (see [73] for an overview). In architecture-based adaptation, a self-adaptive system comprises a *managed system* that is controllable and subject to adaptation, and a *managing system* that performs the adaptations of the managed system. The managed system operates in an *environment* that is non-controllable. The managing system forms a feedback loop that is structured according to the MAPE-K reference model, comprising four functions: Monitor-Analyze-Plan-Execute that share Knowledge [34]. In this mapping study, we analyze primary studies from the perspective of architecture-based adaptation and MAPE-K.

We explain now the basic concepts that we used in the study design. These concepts are based on the process and basic artifacts used in controlled experiments [77]. While we rely on these concepts, we are interested in all papers that apply an experiment in the broad sense, meaning papers that include most of the stages of the process of controlled experiments, explicitly or implicitly. Our particular focus is on technology-oriented experiments that have systems and software elements as subject of the study (in contrast to studies with humans).

An experiment starts with an *idea* for an evaluation, for instance, evaluate a new runtime analysis technique and com-

---

[1]Exemplars published at SEAMS: http://self-adaptive.org/exemplars.

pare it with a state-of-the-art approach. This idea is turned into a hypothesis.[2] The experiment then tests this hypothesis by studying the effect of manipulating one or more independent variables of the studied case. The three types of independent variables are: constants that have a fixed value for the whole experiment, factors that are expected to have an effect on the outcome, and blocking factors that may have an effect but we are not interested in that effect.[3] We use the term *experiment configuration* to refer to the assignment of values to the independent variables[4] and *experiment design* to refer to the set of experiment configurations under study. During an experiment, the effect on the dependent variables caused by different experiment configurations (*i.e.,* selected values for the independent variables) can be measured. Hence, an experiment essentially tests the relationship between the experiment configurations and the outcome, allowing researchers to draw conclusions about the cause-effect relationship to which the approach under study refers for the stated evaluation problem.

The process of an experiment comprises five steps [77]: (1) *experiment scoping* defines the goals of the experiment; (2) *experiment planning* refines the goals to determine the experiment design, which includes selecting a context in which the experiment is carried out, formulating the hypothesis to be tested, selecting the independent and dependent variables, selecting subjects, choosing the experiment configurations, defining how the experiment should be executed and monitored, and evaluating the validity of the results; (3) *experiment operation* prepares and executes the experiment, (4) *analysis & interpretation* analyzes the data collected from the experiment and tests the hypothesis, and (5) *presentation & packaging* presents the results and makes a replication package available.

B. Focus of our Study

This mapping study aims at understanding how evaluations of self-adaptive software systems are performed in studies presented at the International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS). To focus the review, we performed a preliminary analysis of the evaluation methods that were applied in the full papers published at SEAMS between 2011 and 2020. We labelled the evaluation methods according to the following categories: no evaluation, showcase, experiment, review, questionnaire, and proof. A showcase presents results from a *single* experiment configuration [77, p. 75]. An experiment, on the other hand, provides *quantitative* comparative results *for more than one* experiment configuration. A controlled experiment is an experiment that follows a rigorous well-defined process [77]. We found that more than 65% of the examined studies (82 out of 126 full papers) contained at least one experiment. Since the majority of studies use experiments for evaluation, we decided to focus our study on experiments as the evaluation method.

---

[2]Instead of hypotheses, researchers may use research questions, or even informal descriptions to capture the idea of the evaluation.

[3]If an experiment includes a blocking factor whose values create blocking groups, the analysis of the main factors (for which the experiment wants to study the effect) is performed within each blocking group to increase the experiment's precision.

[4]An experiment configuration relaxes Wohlin *et al.*'s definition of a treatment being the assignment of a particular value to one factor [77].

C. Related Work

In the field of self-adaptation, several related efforts pay attention to contributions in the field [14], [26], [38], [40], [44], [49], but do not provide an in-depth study of evaluation aspects. Other related studies do consider evaluation aspects, but they take a specific angle focusing on: claims and evidence in self-adaptive systems [75], [76], quality aspects and metrics [6], [15], [41], [50], [56], [69], and methodology [1], [53], [65]. In contrast, our mapping study targets an in-depth analysis and characterization of the way experimental evaluations have been designed, conducted, analyzed, and packaged.

III. SUMMARY OF THE PROTOCOL

Following the guidelines of [51], we conducted the mapping study with four researchers that jointly developed the protocol. To ensure validity, the protocol was also reviewed by experts in self-adaptation and experimental software engineering. We made the protocol available as part of our replication package.[5]

A. Research Questions

We formulate the goal of our mapping study by using the classic Goal-Question-Metric (GQM) approach [68]:

> *Purpose*: Organize and characterize
> *Issue*: how evaluations of self-adaptive software systems are performed
> *Object*: in research on self-adaptation published at the 10 most recent SEAMS installments
> *Viewpoint*: from a researcher's viewpoint.

We detailed this goal into five concrete research questions that correspond to the five phases of the experiment process.

RQ1: What is the scope of experiments?
RQ2: What is the experimental design?
RQ3: How are experiments operated?
RQ4: How is the experiment data analyzed?
RQ5: How are experiment results packaged?

With RQ1, we aim to understand the purpose and object of evaluations. With RQ2, we want to obtain in-depth insights in independent and dependent variables, experiment configurations, and designs. This will shed light on the complexity and variability of experiments applied for self-adaptive systems. With RQ3, we want to characterize how experiments on self-adaptive system are executed, with particular emphasis on aspects specific to self-adaptive systems such as the distinction between managed and managing system. With RQ4, we want to get insights of how experiment results are analyzed (*e.g.,* using descriptive or inferential statistics). Finally, with RQ5, we want to obtain an overview of whether and how experiment results are made available and packaged for replication.

B. Search Strategy

We examine primary studies published at the main venue on engineering self-adaptive systems—SEAMS. First, there is a normative justification for this focus. Studies presented at

---

[5]https://doi.org/10.5281/zenodo.4622749

SEAMS provide a representative sample of software engineering research of self-adaptive systems. Other studies have also chosen to focus on one specific venue such as ICSE [63], [79]. According to the ACM SIGSOFT Empirical Standards [57], which are currently under development, this is an acceptable deviation to perform secondary studies. Second, there is a qualitative justification. To make a useful and accurate assessment of the features we target in this review, we need relevant data. Based on our combined experience as active members of the SEAMS community, we believe that studies presented at SEAMS provide a source of such relevant data. In light of these two arguments, we acknowledge that our focus may create some degree of bias that we further discuss as a threat to external validity.

C. Inclusion and Exclusion Criteria

We use the following inclusion criteria to select papers:

- IC1: The paper is published at SEAMS between 2011 and 2020 (included). In 2011 SEAMS became a symposium, which increased the level of the evaluations significantly.
- IC2: The paper empirically evaluates an approach by using one or more technology-oriented experiments.

We use the following exclusion criteria:

- EC1: The paper is not a full research paper. These papers typically do not empirically evaluate a new approach.
- EC2: The paper presents a secondary study (*e.g.,* literature review, survey, mapping study) or an overview of the field (*e.g.,* taxonomy, roadmap). These papers do not present and evaluate a novel approach for self-adaptation.

A paper is selected if it meets all of the inclusion criteria and does not meet any exclusion criterion.

D. Data Items

To answer the research questions, we define a set of data items to be extracted from the papers, see Table I. Since the data items refer to a single experiment and a study may contain more than one experiment, we identify all the experiments that are included in a study and then extract the data of each experiment independently. The column "Process Step" shows that our study covers the whole experiment process (see Section II-A) and key aspects that are relevant for technology-oriented experiments as reported at SEAMS.

E. Approach for the Analysis

We tabulate the data that we extract from the primary studies in spreadsheets for processing. We use descriptive statistics to structure and present the quantitative aspects of the data, and comprehensible summaries of the data to answer the research questions. We present results with plots using simple numbers and sometimes means and standard deviations to help understand the results. For the data items F1, F2, F4–F7, and F13, we collect free text and apply coding [71] to capture the essence of the answers. As a concluding step, we produce a schematic overview of the experimental process for self-adaptive systems, allowing us to identify any difference from the "traditional" experiment process (see Section II-A).

IV. Results

A. Demographic Information

From a total of 224 papers presented at SEAMS in the period between 2011 and 2020, we identified 126 full papers and from those, we selected 82 primary studies (65%) after applying the inclusion and exclusion criteria, see Fig. 1. The primary studies reported a total of 140 experiments (1.71 on average, 0.34 std.). The results show that the relative number of full papers that use experiments for evaluation increased from 57% in the period from 2011 to 2015 to 78% in the period from 2016 to 2020 (with even 100% in 2020). At the same time, the average number of experiments per primary study increased from 1.52 in the first period to 1.92 in the second period. These numbers underpin an increasing level of mature evaluations in papers published at SEAMS over time.
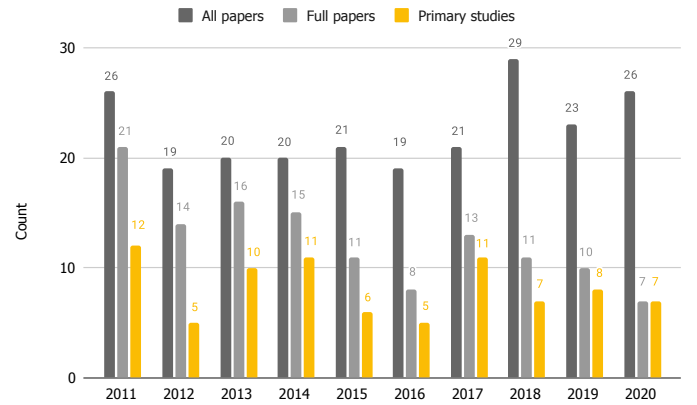

Fig. 1: Overview of selected primary studies per year.

B. What is the Scope of Experiments?

To answer RQ1, we collected the data about the targets of the evaluations (F1), and the objectives of evaluations (F2).

Fig. 2 plots the counts for the targets of the evaluations (F1) reported in the primary studies. In 50 studies (61%), the evaluation targeted a new integrated adaptation approach that covers the full feedback look. For instance, Derakhshanmanesh *et al.* [16] present an adaptation framework that uses graph-based models throughout the feedback loop. Next, 16 studies (20%) evaluated a new learning method. For instance, Duarte *et al.* [18] contribute a method for learning linear models that capture non-deterministic impacts of adaptation. Notably, the numbers for new learning methods increased from five in the period from 2011 to 2015 to 11 in the period from 2016 to 2020. The remaining studies focused on evaluating new approaches for distinct MAPE-K stages. Among these, 11 studies targeted a planning method and ten studies targeted an analysis method. Only one study targeted a new monitoring approach [42] and one other study targeted an execution approach [23]. Independently of the evaluation targets, 79 of all 140 experiments (56%) reported in the primary studies used the full feedback loop, while 61 experiments (44%) considered only a part of the feedback loop in the evaluation.

Fig. 3 shows the results of the evaluation objectives (F2) used in the experiments. We extracted 165 individual evaluation objectives from the 140 experiments: 116 experiments

TABLE I: Extracted data items.

| ID	Item	Use	Process Step	Explanation
F1	Target of evaluation	RQ1	Experiment scoping	The main element that is subject of evaluation, incl. the whole feedback loop and methods for distinct MAPE-K stages and learning.
F2	Objectives of evaluation	RQ1	Experiment scoping	The aspects of the proposed approach that are evaluated, mentioned explicitly or implicitly.
F3	Formulation of evaluation problem	RQ2	Experiment planning	Captures whether there is an explicit formulation of the evaluation problem by either research questions or hypotheses.
F4	Constants	RQ2	Experiment planning	The names of the variables that are constant across the experiment.
F5	Blocking factors	RQ2	Experiment planning	The names of the variables that are used to create experiment blocks, but without interesting effect [77, p. 94].
F6	Factors	RQ2	Experiment planning	The names of the variables that change across experiment configurations.
F7	Dependent variables	RQ2	Experiment planning	The names of the variables that measure the effect of an experiment configuration, also called "response variables" [77, p. 74]).
F8	Counts experiment variables	RQ2	Experiment planning	The number of values of independent and dependent variables used in experiments (referring to F4, F5, F6, and F7).
F9	Design type	RQ2	Experiment planning	The design type used in the experiment, following the standard design types described by Wohlin *et al.* [77, p. 95].
F10	Managed system name	RQ3	Experiment operation	The name of the managed system, if any. The managed system may be a SEAMS artifact, see http://self-adaptive.org/exemplars.
F11	Nature of managed system	RQ3	Experiment operation	The type of managed system used in the evaluation, incl. model, simulated/emulated, real implementation, and real-world application.
F12	Data provenance	RQ3	Experiment operation	Source of data related to the users or the environment of the managed system, incl. synthetic data, emulated data, and real-world data.
F13	Uncertainty	RQ3	Experiment operation	The way uncertainty is represented in the experiment. This type of uncertainty can create the need for self-adaptation.
F14	Type of analysis	RQ4	Analysis & interpretation	The type of analysis that is performed on the experiment results, incl. none, exposition (narrative), descriptive statistics, and statistical tests.
F15	Answer to evaluation problem	RQ4	Analysis & interpretation	Whether there is an explicit answer to research questions or hypotheses.
F16	Threats to Validity	RQ4	Analysis & interpretation	The types of threats to validity mentioned (in a dedicated section/subsection or paragraph), if any.
F17	Results available	RQ5	Presentation & packaging	Captures whether evaluation results are available (*e.g.,* via a URL).
F18	Degree of reproducibility	RQ5	Presentation & packaging	Captures whether the implementation of the approach is available or the full replication package is available (*e.g.,* via a URL).
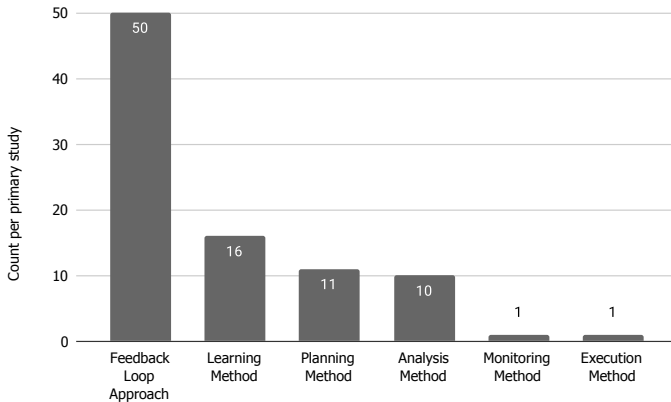
Fig. 2: Count of evaluation targets (F1) per primary study.

(83%) had one evaluation objective, 23 (16%) had two objectives, and one had three objectives. The top-reported evaluation objective is effectiveness that was used 75 times (45% of 165), followed by learning ability (used 34 times, 21%) and time efficiency (used 24 times, 15%). As examples, Jamshidi *et al.* evaluate the effectiveness in terms of the number of completed robot missions [32], while Nikravesh *et al.* evaluate the learning ability by assessing accuracy of different workload predictors [45]. Sousa *et al.* evaluate the time efficiency of planning by measuring the time to find

a valid configuration [64], while Shin *et al.* evaluate the scalability of a search-based adaptation approach in terms of execution time with increasing network size [62].

Fig. 4 maps the evaluation targets (F1) to the objectives (F2). The results show that effectiveness is used as evaluation objective for all types of evaluation targets. New feedback loop approaches are mostly evaluated for effectiveness (46 experiments) and time efficiency (17 experiments). Not surprisingly, learning ability is the top evaluation objective for new learning methods (in 31 experiments). Scalability is used as evaluation criterion for four of the six evaluation targets (not for the single new proposed execution [23] and monitoring methods [42]).

> **Answer to RQ1: What is the Scope of Experiments?** The main evaluation target of experiments in self-adaptive systems is a new integrated feedback loop approach with effectiveness and time efficiency as main evaluation objectives. Recently, we observe a rapid increase in experiments that focus on new learning approaches evaluated for their ability to learn and effectiveness.

C. What is the Experimental Design?

To answer RQ2, we collected data about the formulation of the evaluation problem (F3), the independent variables (*i.e.,* constants (F4), blocking factors (F5), and factors (F6)), the dependent variables (F7), the counts of values of the different types of variables (F8), and the design type (F9), see Table I.
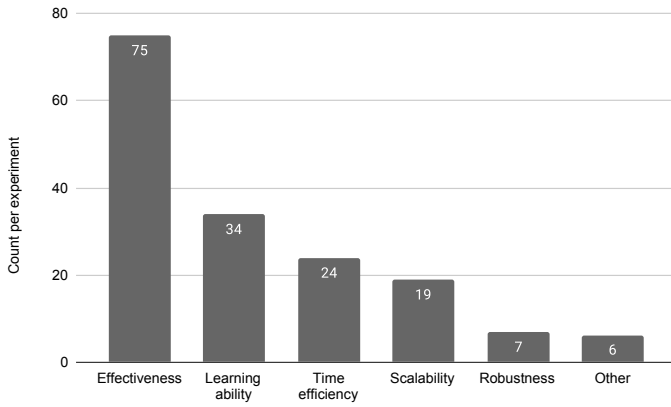
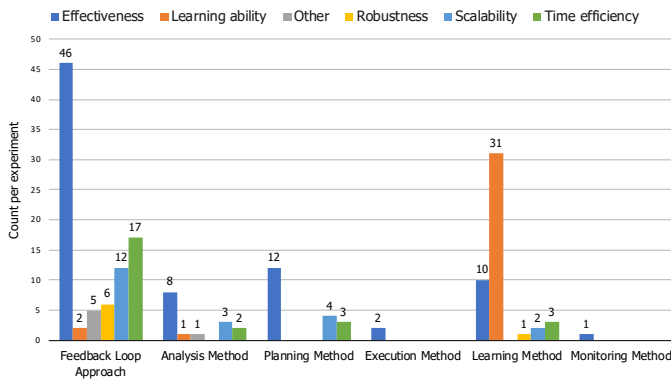Fig. 3: Count of evaluation objectives (F2) per experiment.

Fig. 4: Mapping of evaluation targets (F1) to objectives (F2).

Fig. 5: Count of independent variables (F4, F5, F6).

Only 28 studies (34%) provide a well-defined formulation of the evaluation problem (F3), of which 21 (26%) use research questions and 7 (8%) use hypotheses. For example, Jamshidi *et al.* specify three research questions (on accuracy, effectiveness, and robustness) that guide their evaluation [31], while Fredericks uses null hypotheses to compare the proposed approach to a baseline (on effectiveness of generating adversary environments and effectiveness of adaptation) [21]. The remaining 54 primary studies (66%) provide an informal description of the evaluation problem.[6] For example, Pournaras *et al.* describe the goal of their evaluation in an informal way [54]. Remarkably, this result resembles those of an early survey of primary studies of SEAMS before the year 2012 [76], suggesting little progress in formulating clearly defined research problems in studies presented at SEAMS.

Fig. 5 gives an overview of the overall count of independent variables (F4, F5, F6) in the experiments (see Section II-A for a description of the different types of independent variables). From all the experiments reported in the primary studies we extracted 141 constants (avg. per experiment 1.01, std. 0.95). "Load profile" is the constant with the highest number of occurrences (used in 26 experiments, *e.g.,* [17], [20], [29]); other examples are "number of nodes" [12], "number of sensors" [25] and "learning/optimization hyperparameters" [45], [48]. From all experiments, we extracted 95 blocking factors
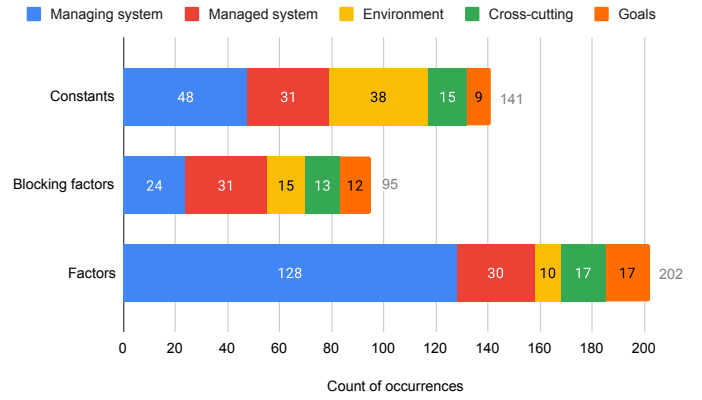
(avg. per experiment 0.68, std. 0.91). For example, "deployment environment" was used to block the analysis of elasticity configurations in two settings: private and public cloud [28]. Finally, we extracted in total 202 factors (avg. per experiment 1.44, std. 0.85). For example, "assurance approach" in [21] took two values that correspond to the proposed approach (genetic algorithm) and a baseline (random search) that are evaluated against adversarial environments of the system.

The results show that 200 of a total of 438 independent variables (46%) relate to the managing system. Specifically, 128 of a total of 202 factors (28%) relate to the managing system, indicating that experiments in self-adaptive systems target prominently the evaluation of new approaches and methods of the managing system. On the other hand, 92 independent variables (21%) relate to the managed system and 63 relate to the environment (14%), the latter are mostly constants in the experiments. Notably, only 38 independent variables (9%) relate to system goals (17 of these are factors). These figures suggest a relatively low interest in considering the impact of goals in the evaluation of new approaches for self-adaptive systems. Finally, the group "Cross-cutting" refers to independent variables that cross-cut at least two elements of a self-adaptive system (managing system, managed system, environment, goals). We extracted 45 such independent variables (10%). Among these, the most frequent combination is a variable that cross-cuts managed system and goals representing a scenario that warrants adaptation. For example, Shevtsov *et al.* [60] uses a scenario that is defined by a set of sensors that need to perform a monitoring task with maximum measurement accuracy while being exposed to failures.

Fig. 6 shows the results for the dependent variables (F7). We identified five classes of dependent variables from a total count of 267 concrete variables used in the experiments (on average, 1.91 variables per experiment; 1.10 std.). The dominant dependent variable is "Time behavior"[7] that was used 127 times (48% of the total count). As an example, the response time of an online news service (ZNN, a popular SEAMS artifact) was measured in [2]. Other frequently used dependent variables

---

[6]With informal description we mean the evaluation problem is described with some general words or is only provided implicitly.

[7]These variables measure a time-related property. The most prominent variables are response time (31 times used) and processing time (29 times).
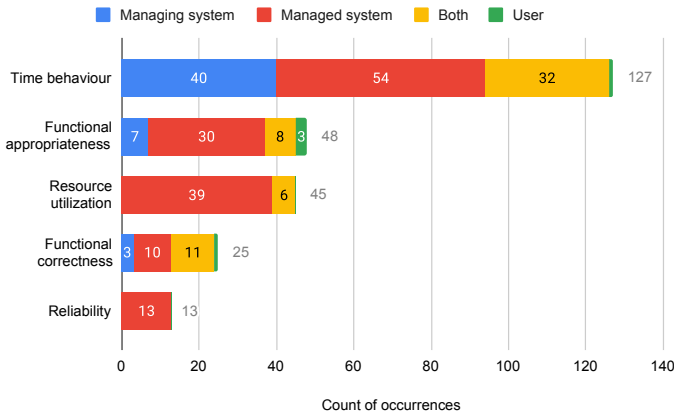
Fig. 6: Count of dependent variables (F7). Out of the total of 267 occurrences, 9 belong to smaller categories not depicted.

Fig. 7: Standard design types used in experiments (F9).

are "Functional appropriateness"[8] (48 times; 18% of the total count) and "Resource utilization" (45 times; 17% of the total count). For example, "distance scanned" was used to assess the functional appropriateness of the proposed solution in [60], while the number of servers was used to assess the resource consumption in [28]. Notably, we found only 13 concrete variables related to "Reliability."[9] For example, packed loss is used to assess reliability in [67]. The dependent variables refer mostly to the managed system (146 times, 55% of the total count) followed by both managing and managed system (57 times, 21%) and managing system (50 times, 19%).

Table II summarizes the average numbers of different types of variables (F8) used per experiment.

TABLE II: Average numbers of variables per experiment.

Variable	Avg	Std
Constants	1.01	0.95
Blocking factors	0.68	0.91
Factors	1.44	0.85
Dependent variables	1.91	1.10

While these numbers give us an insight in the variables used in the experimental design, we also extracted data about the use of standard design types (F9) to get a better view of the concrete design types used in experiments in self-adaptive systems.[10] Fig. 7 shows the results. Out of all 140 experiments, 55 experiments (39%) use a standard design type. The most frequently used standard design type is "One factor with more than two values" (32 experiments, 23%). For example, the experiment design in [9] involves one factor ("managing system") with three values corresponding to using built-in adaptation mechanisms, using architecture-based adaptation (Rainbow) with default adaptation strategies,

[8]These dependent variables measure the suitability of a new approach from a functional viewpoint. Examples are the degree that goals are satisfied and the degree of financial profit obtained from applying a new approach.

[9]We counted the variables that explicitly refer to reliability, or are clearly connected such as packet loss in a network. However, variables of other classes may indirectly relate to reliability. E.g., a variable that measures functional correctness may be important to achieve a required level of reliability.

[10]We use the four standard design types for experimentation in software engineering described by Wohlin *et al.* [77, p. 95].
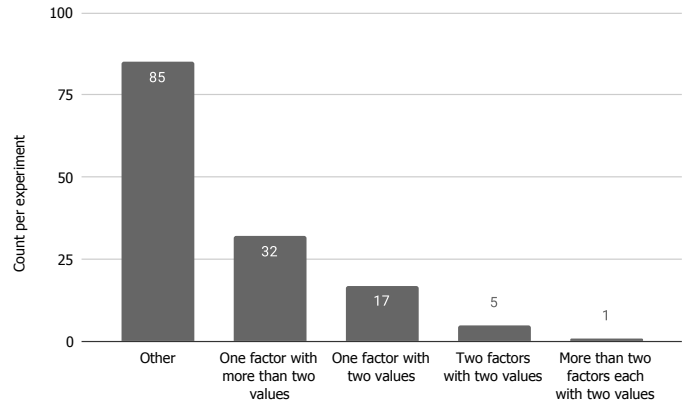
and using Rainbow with improved adaptation strategies. Of the 55 experiments that use a standard design, 49 (35% of all experiments) use one factor with two or more values. However, a majority of 85 experiments (61%) do not use a standard design type. These experiments use a design with different combinations of factors and values for these factors. For instance, the experiment by Kistowski *et al.* to evaluate load extraction methods has one factor and three blocking factors, generating a total of 96 experiment configurations [36].

To get further insight in the concrete designs of experiments in self-adaptive systems, we combined the data collected for the evaluation objectives (Fig. 3), independent variables (Fig. 5), and dependent variables (Fig. 6). This allowed us to identify a number of patterns for different evaluation objectives that map independent to dependent variables, see Table III.[11]

The pattern for effectiveness applies two methods of a managing system combined with constants, and measures the effect on resource utilization or time behavior of a managed system (13 instances). For example, Barna *et al.* [4] evaluate the effectiveness of mitigating DoS attacks by comparing two mitigation methods based on measured CPU utilization and response time of the managed system. The pattern for scalability applies a single method of a managing system on different variations of a managed system, and measures the scalability for time behavior of the managing system or functional appropriateness of both the managed and managing system (8 instances). For example, Incerto *et al.* [30] evaluate the scalability of an SMT-backed planning approach in terms of computation time under increasing numbers of servers in the managed system. The pattern for time efficiency applies more than two methods of a managing system combined with constants, and measures the effect on the time behavior of both managing and managed system or just the managing system (4 instances). For example, Kumar *et al.* [39] compare the time efficiency of four self-adapting service composition approaches by measuring the planning time. Finally, the pattern for learning ability applies two methods of the managing system combined with multiple parameter settings of the man-

[11]As notation to describe a combination of independent variables we use: "$var_1 (n_{var_1}) \times var_2 (n_{var_2}) \times ...$", where var_1 is the variable and n_{var_i} is the number of values of var_i. We use an asterisk as a wild card for var_i. Note that $n_{var_i} = 1$ if var_i is a constant.

TABLE III: Identified patterns for different objectives that map independent to dependent variables.

| Objective	Independent variables	#	Dependent variables (top two)
Effectiveness	Managing-Method (2) × * (1)	13	Resource utilization [managed], Time behaviour [managed]
Scalability	Managing-Method (1) × Managed-Variation (>2) × * (1)	7	Time behaviour [managing], Functional appropriateness [both]
Time efficiency	Managing-Method (>2) × * (1)	4	Time behaviour [both], Time behaviour [managing]
Learning ability	Managing-Method (2) × Managing-Parameter (>2) × * (1)	3	Time behaviour [managed], Resource utilization [managed]
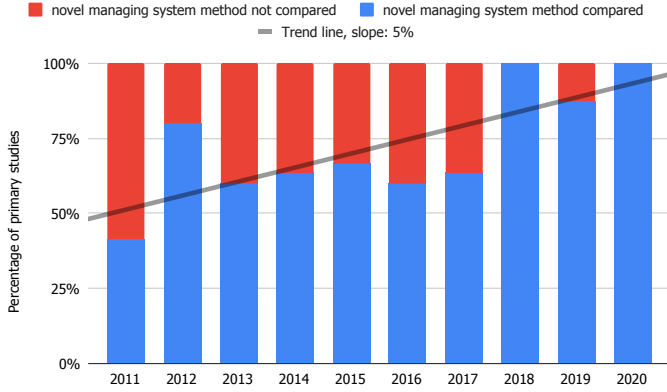
Fig. 8: Percentage of primary studies per year that do (blue) and do not (red) compare a novel managing system approach with at least one other approach.

Fig. 9: Named managed systems (F10) used in at least two primary studies (43 studies provided no specific name).

aging system (typically related to learning hyperparameters) and constants. It measures the effect on the learning ability (*e.g.,* in terms of accuracy or correlation) for the time behavior or functional correctness of a managed system (3 instances). For example, Duarte *et al.* [18] evaluate the accuracy of two learning methods that predict response time of the managed system, configured with different sizes of training data.

To conclude, we measured the percentage of primary studies that do compare a newly proposed managing system approach with at least one other approach, which can for instance be a state-of-the-art approach, primitive adaptation, or the theoretical optimal. Fig. 8 shows the results over the years. The graph clearly illustrates that researchers have increasingly compared new contributions with existing approaches or other baselines. By applying regression on the data, we identified an average yearly increase of 4.67% (rounded 5%) over the 10 years (from 51.3% in 2011 to 93.3% in 2020). This confirms an increasing maturity in the evaluations of new contributions in self-adaptive systems presented at SEAMS, which has now reached an excellent level.

> **Answer to RQ2: What is the Experimental Design?** Only one out of three studies provides a well-defined formulation of the evaluation problem, mostly using research questions. Experiments use independent variables for all parts of self-adaptive systems, with most factors related to the managing system. The dominant types of dependent variables are time behavior, functional behavior, and resource utilization, typically of managed systems. New contributions are increasingly compared with other approaches.

D. How are Experiments Operated?

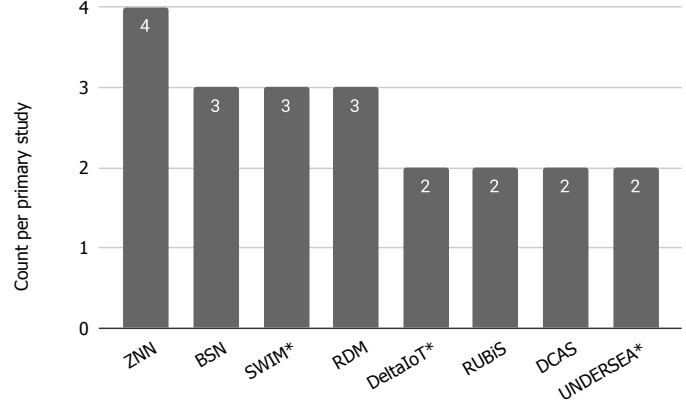To answer RQ3, we collected data about the managed system and whether an artifact was used (F10), its nature

(F11), data provenance, *i.e.,* sources of data related to users or environment (F12), and representations of uncertainty (F13).

Fig. 9 shows the counts of managed systems (F10) that were used in experiments reported in at least two primary studies. The managed systems marked with an asterisk are formally approved SEAMS artifacts. Of the total 82 primary studies, 39 (46%) provided a clearly described name for the managed system. The remaining 43 studies provided no specific name for the managed system. Not surprisingly, ZNN has been used most frequently, BSN (Body Sensor Network), SWIM, and RDM (Remote Data Mirroring) are each used in three studies, and DeltaIoT, RUBiS (Rice University Bidding System), DCAS (Data Acquisition and Control Service) and UNDERSEA are each used in two studies. In total, 10 primary studies used at least one artifact in their evaluation in the period from 2016 to 2020 (*i.e.,* 26% of the primary studies in this period).[12] This result underpins the usefulness of the SEAMS artifacts in the evaluation of new contributions.

Fig. 10 shows the different types of managed systems (F11) used in the primary studies. Thirty-two studies (39%) used a simulation or emulation of a managed system. For example, Gerasimou *et al.* [25] use UNDERSEA, a simulator of unmanned underwater vehicles. Twenty-one studies (38%) used a model to represent a managed system. For example, Incerto *et al.* [30] represent a three-tier managed system as a Queuing Network to evaluate performance adaptation. A real implementation of a managed system was used in 21 studies (26%). This category includes implemented systems based on a model of a real application. For example, Barna *et al.* [3] use LEGIS, a distributed navigation service based on Google maps. On the other hand, in eight studies (10%) the managed

---

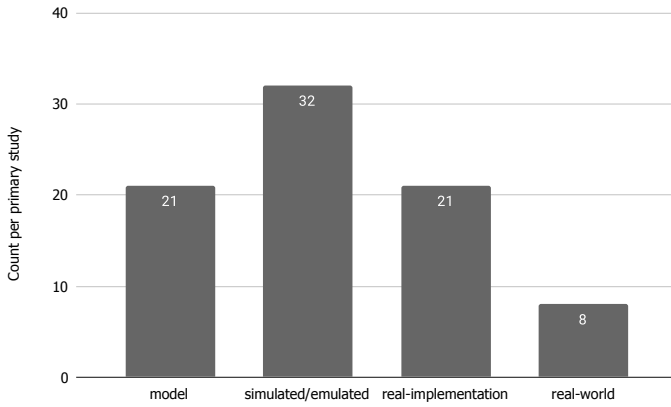[12]The SEAMS Call for Artifacts was introduced in 2015.

Fig. 10: Types of managed systems used per study (F11).

Fig. 11: Representation of uncertainty per study (F13).

system was a real-world application. This category refers to systems that have been used in practice with real users (but not necessarily for the experiments). An example are open-source mobile apps that are used in [58]. These results show that a relevant number of experiments rely on real implementations of managed systems, yet with opportunities to further improve on the use of real-world systems in experiments.

The results for data provenance (F12) show that a majority of 99 experiments (71%) use synthetic data to represent users or the environment. For example, Guerriero *et al.* [27] randomly generate consumer transactions, while Moreno *et al.* [43] generate server boot latencies from normal distributions. Twenty-eight experiments (20%) use emulated data to represent users or the environment. For example, Shin *et al.* [62] emulate a data traffic profile specified by their industry partner to create load on the managed network.

Notably, only a small fraction of the experiments (13, *i.e.,* 9%) use real-world data to represent users or the environment. An example is [36] that uses real-world workload traces of the Internet Traffic Archive, Bibsonomy, and Wikipedia to extract load profiles. These results show that there is room for improvement to represent users and the environment more realistically in experiments of self-adaptive systems.

Fig. 11 shows the results we obtained for the representation of uncertainty (F13). In total, we extracted 132 representations of uncertainty used in the experiments of the primary studies.[13] We could group these 132 representations in four types. The most frequently used type is *uncertainty in the context* that was used 68 times (52% of all represented uncertainties). As an example, Jamshidi *et al.* [32] consider the uncertainty of having obstacles appearing in the robot's environment. *Uncertainty in the system* was used 35 times (27%).[14] For example, Incerto *et al.* [30] address uncertainty of the system in terms of random faults of servers and network links. Only a few studies *considered uncertainty of goals* (20 studies, 15%) and humans (9 studies, 7%). For example, Camara *et al.* [10] randomly assign missions to robots, while Tun *et al.* [66]
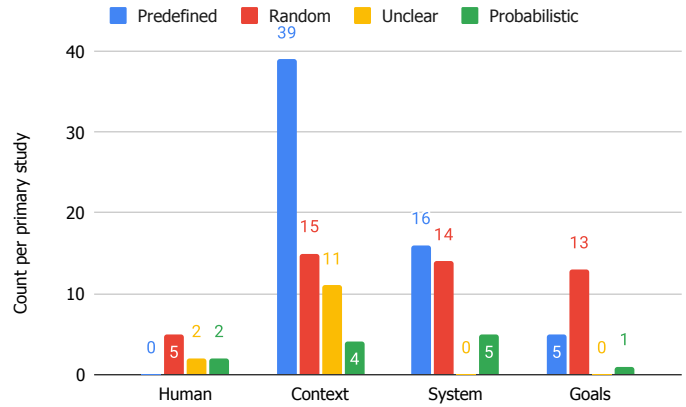
randomly select invitees (users) for sharing files.

Uncertainty in the experiments is mostly represented by predefined values (46% of all uncertainty representations). Such values can be based on common properties or characteristics of the represented topic. For example, Zanardi and Capra [78] consider different predefined adaptation thresholds to model the uncertainty related to goals. Uncertainties (35%) were represented by randomly selected values 64 times (*e.g.,* [66]). Only 12 uncertainties (9%) were represented by probabilistically selected values (*e.g.,* [35], [43]).

The results of F13 show that uncertainty is commonly considered in experiments of self-adaptive systems. Yet, the emphasis is on uncertainty in the context and system. There is room for improvement by putting more emphasis of representing uncertainties related to goals and humans in experiments.

> **Answer to RQ3: How are Experiments Operated?** Artifacts are increasingly used in experiments of self-adaptive systems. The managed system is mostly simulated or emulated. Yet, one on three studies uses system implementations, but the number of real-world systems or prototypes of such systems remains relatively low. Most data of users and the environment used in experiments is synthetically generated. Studies commonly consider uncertainties in the context and the system, mostly represented by selected values (rather than randomly or probabilistically). Uncertainties related to goals and humans are not frequently considered.

E. How is the Experiment Data Analyzed?

To answer RQ4, we collected data about the types of analysis applied (F14), the answers provided for the evaluation problem (F15) and the discussion of treats to validity (F16).

The results for the type of analysis (F14) performed in the experiments show that a majority of 62 studies (44%) used some form of exposition or narrative to analyze and discuss the experiment results. For example, Weisenburger *et al.* [72] plot the timeseries of latency and bandwidth obtained by their approach and the baseline under different settings and discuss the observed behavior. Fifty-nine studies (42%) used descriptive statistics to show or summarize data in a meaningful way (*e.g.,* using tables, graphs and charts), which allows identifying patterns that might emerge from the data. As an example, Sousa *et al.* [64] analyze their experiment by calculating statistics (average, standard deviation, maximum, minimum)

---

[13]Note that if two experiments of the same primary study used the same uncertainty representation, this was counted once.

[14]Uncertainty of the system relates to the managed system, either represented in the managed system itself or in a runtime model of the managed system used by the managing system.

of execution times over 12 runs. Finally, 19 studies (14%) used statistical tests to analyze the data of the experiment and draw conclusions. A statistical test provides a systematic mechanism for making a quantitative decision about the outcome of an experiment; for instance to determine whether there is enough evidence to reject a null hypothesis. For example, Fredericks *et al.* [22] define a hypothesis to test statistically whether a difference exists between the result of their proposed approach and a baseline. We also extracted data about whether the choice for a statistical test was motivated and found that only 12 of the 19 studies do so. The results show that a substantial part of the studies still apply informal approaches for the analysis of data of experiments. There is room for applying more rigorous methods of analysis of data obtained from experiments in self-adaptive systems.

Of the 28 studies that formulated evaluation problems using either research questions or hypothesis, only 19 (68%) provided an explicit answer to the evaluation problem. For example, Chen [11] summarizes the key findings of each of the specified four research questions, while Fredericks *et al.* [22] explain the rejection of the specified null hypothesis based on a statistical test. The numbers underpin a need for improvement on reporting research findings from experiments, in particular providing answers to the research questions under study.

Fig. 12 shows the results for whether and how threats to validity of experiments (F16) are discussed in primary studies. In total only 31 primary studies (38% of all primary studies) provided some discussion of validity threats. Seventeen of these studies (55% of the 31 studies that discuss validity threats) provide an informal discussion of validity threats without referring to any particular types of threats. As an example, Sousa *et al.* [64] discuss the limitations of their experiments in a separate section in an informal way. The most reported validity threats are internal and external validity, both discussed in 14 studies (45% of the 31 studies that discuss validity threats). For example, Jamshidi *et al.* [33] discuss internal and external validity and their attempt to mitigate these threats, and discuss remaining limitations. Of these 14 studies, seven also discuss construct validity. As an illustration, Kumar *et al.* [39] mention construct validity threats related to the employed metric and evaluation methods, and how they mitigate them. Only one study [55] discusses reliability pointing out that reproducing the results of the study may be affected by randomness included in the simulation setup. Acknowledging and discussing validity threats is key for future research as they pinpoint potential issues with the experimental design and the causal relationships and generalization of results. Hence, there is room for improvement on discussing validity threats for experiments of self-adaptive systems.

> **Answer to RQ4: How is the Experiment Data Analyzed?** A small half of the primary studies use an informal approach to analyze the data obtained from experiments. Another small half uses descriptive statistics and only a fraction of the studies uses statistical tests. Only a limited number of the primary studies provide explicit answers to the research problems they tackle. Explicitly discussing threats to validity is not common practice in experimental research of self-adaptive systems.
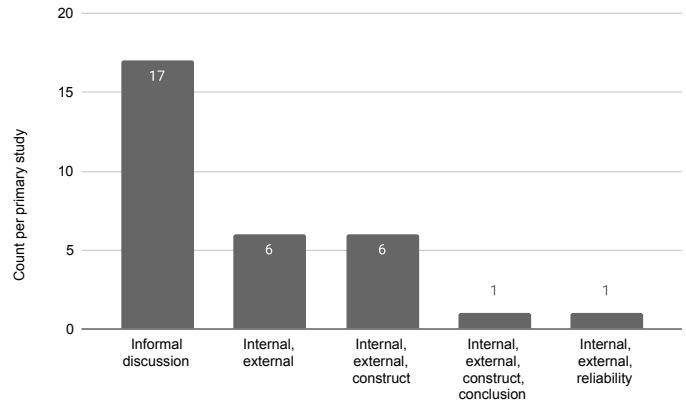

Fig. 12: Discussion of threats to validity per study (F16).

F. How are Experiment Results Packaged?

To answer RQ5, we collected data about the availability of the experimental results (F17), and the degree of reproducibility (F18) of experiments.

Only 11 of the primary studies (13%) made the evaluation results of their experiments publicly available (F17). Examples of studies that provide evaluation results are [30], [62], where results are made available with a replication package. Making experiment data public allows for verifying findings and experimental reuse, and lowers the barriers to meta-studies.[15] Hence, there is room for improvement here, but this is a general problem and applies also to other research fields than self-adaptive systems (*e.g.,* [52]).

The results we obtained for the degree of reproducibility (F18) of experiments reported at SEAMS echo those of F17. Only nine studies (11%) provide a full replication package, while 14 other studies (17%) provide the code used in experiments. Although it is a foundation of science, replication is a recurrent issue in empirical software engineering in general and this applies also to software engineering of self-adaptive systems. The results of this study show a slight improvement in terms of reproducibility compared to the results of the earlier study [76] that looked at research presented at SEAMS before 2012. There, 14% of the studies provided partial material for repeatability and only 2% provide a full replication package.

All in all, there remains substantial room for further improvement on providing replication packages facilitating cross-validation and comparison across studies.

> **Answer to RQ5: How are the Experiment Results Packaged?** Only a small fraction of the primary studies make the data of their experiment publicly available. Similarly, the degree of reproducibility remains low, only one study on ten offers a full replication package to the community.

V. DISCUSSION

We start this section with summarizing insights on the specifics of experiments in self-adaptive systems. Then we offer suggestions to improve future experiments in self-adaptive systems. Finally, we discuss threats to validity of this study.

---

[15]www.elsevier.com/connect/should-research-data-be-publicly-available

A. Experiments in Self-adaptive Systems vs Other Systems

This mapping study provides a number of insights on the specifics of experiments in self-adaptive systems compared to other systems. In Fig. 13, we list these insights based on the five steps of the process of an experiment [77].
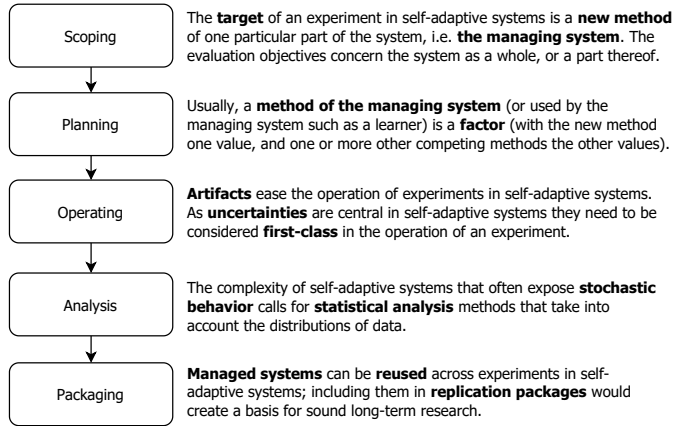

Scoping	The **target** of an experiment in self-adaptive systems is a **new method** of one particular part of the system, i.e. **the managing system**. The evaluation objectives concern the system as a whole, or a part thereof.
Planning	Usually, a **method of the managing system** (or used by the managing system such as a learner) is a **factor** (with the new method one value, and one or more other competing methods the other values).
Operating	**Artifacts** ease the operation of experiments in self-adaptive systems. As **uncertainties** are central in self-adaptive systems they need to be considered **first-class** in the operation of an experiment.
Analysis	The complexity of self-adaptive systems that often expose **stochastic behavior** calls for **statistical analysis** methods that take into account the distributions of data.
Packaging	**Managed systems** can be **reused** across experiments in self-adaptive systems; including them in **replication packages** would create a basis for sound long-term research.

Fig. 13: Main insights on the process of experiments.

B. Suggestions for Improving Future Experiments

In what follows, we offer a number of suggestions that we obtained from this study as impetus to improve future experiments in self-adaptive systems. Experiments on new methods for the monitor and execute stages of MAPE-K feedback loops require attention. Security, privacy, humans in the loop, and ethical considerations are widely regarded as key concerns of modern software systems but they are not well studied in our field, hence they deserve attention. We can improve on better formulating the evaluation problems we tackle. We often use different types of design compared to experiments in traditional software systems; this may either point to a lack of maturity or the need for designs specific to experimentation in self-adaptive systems; this deserves further investigation. We observe an increasing trend in the use of real-world managed systems; experiments would benefit from pushing this trend further. Uncertainty is a complex phenomenon but essential to self-adaptation; there are plenty of opportunities to enhance on how we represent uncertainties in experiments. There is substantial room to improve on the analysis of experimental data; in particular by applying statistical techniques. Experiments in self-adaptive systems would benefit from a more rigorous description of different types of validity threats, and making replication packages available for the community. We hope that these empirically grounded suggestions will help the community to improve the way we evaluate self-adaptive systems in the future.

C. Threats to Validity

While following a systematic approach based on a protocol, this study has some possible threats to validity.

Internal validity refers to the extent to which a causal conclusion can be made based on the study. Determining whether a paper contained an experiment was sometimes not easy as some information may be implicit. In addition, the extraction of detailed information about experiments, in particular identifying different types of variables was also not always easy. To mitigate this threat, we took the following measures. (i) All papers where checked for inclusion or exclusion by at least two researchers. (ii) The primary studies were split in three parts; for the first part (10% of the studies) data was extracted in parallel by two researchers and decisions were made based on consensus; in case of conflicts a third researcher was consulted to make a decision; for the two other parts, data was extracted by one researcher and crosschecked by another (and if needed by a third). (iii) Data analysis was jointly done by all researchers in collaboration.

External validity refers to the extent to which the findings can be generalized. By considering only full papers presented at SEAMS, we obviously can only draw conclusions for this venue. However, as argued in the summary of the protocol (Section III), the papers presented at SEAMS provide a representative sample of software engineering research of self-adaptive systems. Furthermore, the draft of the ACM SIGSOFT Empirical Standards consider focusing on a single venue an acceptable deviation of performing secondary studies [57]. Nevertheless, to strengthen the validity of our study, a broader search for primary studies would enhance validity.

Construct validity refers to the extent to which we obtained the right measure and whether we defined the right scope in relation to the topic of our study. There is threat that the reporting of experiments in some papers may not be of sufficient quality. However, since SEAMS became a symposium in 2011, we believe that papers that were accepted as full papers provide a sufficiently good quality of reporting. We acknowledge that an additional quality check based on the quality criteria for reporting studies as *e.g.,* used in [19], [61] may help improving the validity of our results.

Reliability refers to the extent to which we can ensure that our results are the same if our study would be conducted again. An obvious threat is a potential bias of the researchers involved in our study, in particular when collecting and analyzing data of primary studies. To address this threat, we used a protocol that we carefully followed. In addition, we have made all the material of the study available for other researchers.

VI. Conclusions

This mapping study aimed at answering the question "How do we evaluate self-adaptive software systems?" with a focus on technology-oriented experiments presented at SEAMS from 2011 till 2020. Results show that experiments in self-adaptive systems do follow standard practice on empirical research in software engineering, but at the same time also have some specifics that deserve special attention across the stages of the experimental process. These specifics are essentially based on characteristics of self-adaptive systems, such as the evaluation target that is associated with the managing system, and the presence of uncertainties that require attention in experiment design and analysis. Our study allowed us to provide a number of suggestions for improving experimental evaluations of self-adaptive systems. We hope that these suggestions and the results obtained from our study trigger reflection in the community on doing future experiments with even more maturity.

REFERENCES

[1] Jesper Andersson, Luciano Baresi, Nelly Bencomo, Rogério de Lemos, Alessandra Gorla, Paola Inverardi, and Thomas Vogel. Software engineering processes for self-adaptive systems. In *Software Engineering for Self-Adaptive Systems II*, pages 51–75. Springer, 2013.

[2] Konstantinos Angelopoulos, Vítor E. Silva Souza, and João Pimentel. Requirements and architectural approaches to adaptive software systems: A comparative study. In *International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS. IEEE, 2013.

[3] Cornel Barna, Hamzeh Khazaei, Marios Fokaefs, and Marin Litoiu. Delivering elastic containerized cloud applications to enable devops. In *12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS, page 65–75. IEEE, 2017.

[4] Cornel Barna, Mark Shtern, Michael Smit, Vassilios Tzerpos, and Marin Litoiu. Model-based adaptive dos attack mitigation. In *7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS, page 119–128. IEEE, 2012.

[5] Stephen M. Blackburn, Amer Diwan, Matthias Hauswirth, Peter F. Sweeney, et al. The truth, the whole truth, and nothing but the truth: A pragmatic guide to assessing empirical evaluations. *ACM Trans. Program. Lang. Syst.*, 38(4), 2016.

[6] Yuriy Brun. Improving impact of self-adaptation and self-management research through evaluation methodology. In *ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS, page 1–9. ACM, 2010.

[7] R. Calinescu, R. Mirandola, D. Perez-Palacin, and D. Weyns. Understanding uncertainty in self-adaptive systems. In *International Conference on Autonomic Computing and Self-Organising Systems*, 2020.

[8] R. Calinescu, D. Weyns, S. Gerasimou, M. U. Iftikhar, I. Habli, and T. Kelly. Engineering trustworthy self-adaptive software with dynamic assurance cases. *IEEE Transactions on Software Engineering*, 44(11):1039–1069, 2018.

[9] Javier Cámara, Pedro Correia, Rogério de Lemos, David Garlan, Pedro Gomes, Bradley Schmerl, and Rafael Ventura. Evolving an adaptive industrial software system to use architecture-based self-adaptation. In *8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS, page 13–22. IEEE, 2013.

[10] Javier Cámara, Bradley Schmerl, and David Garlan. Software architecture and task plan co-adaptation for mobile service robots. In *15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS, page 125–136. ACM, 2020.

[11] Tao Chen. All versus one: An empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software. In *14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS. IEEE, 2019.

[12] Tao Chen and Rami Bahsoon. Symbiotic and sensitivity-aware architecture for globally-optimal benefit in self-adaptive cloud. In *9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS, page 85–94. ACM, 2014.

[13] B. Cheng et al. Software engineering for self-adaptive systems: A research roadmap. In *Software Engineering for Self-Adaptive Systems*, pages 1–26. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[14] Mirko D'Angelo, Simos Gerasimou, Sona Ghahremani, Johannes Grohmann, Ingrid Nunes, Evangelos Pournaras, and Sven Tomforde. On learning in collective self-adaptive systems: State of practice and a 3d framework. In *14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS. IEEE, 2019.

[15] Amanda Oliveira de Sousa, Carla I. M. Bezerra, Rossana M. C. Andrade, and José M. S. M. Filho. Quality evaluation of self-adaptive systems: Challenges and opportunities. In *XXXIII Brazilian Symposium on Software Engineering*, SBES 2019, page 213–218. ACM, 2019.

[16] Mahdi Derakhshanmanesh, Mehdi Amoui, Greg O'Grady, Jürgen Ebert, and Ladan Tahvildari. Graf: Graph-based runtime adaptation framework. In *6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS, page 128–137. ACM, 2011.

[17] Antinisca Di Marco, Paola Inverardi, and Romina Spalazzese. Synthesizing self-adaptive connectors meeting functional and performance concerns. In *8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS. IEEE, 2013.

[18] Francisco Duarte, Richard Gil, Paolo Romano, Antónia Lopes, and Luís Rodrigues. Learning non-deterministic impact models for adaptation. In *13th International Conference on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS, page 196–205. ACM, 2018.

[19] Tore Dybå and Torgeir Dingsøyr. Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(9):833 – 859, 2008.

[20] Marios Fokaefs, Cornel Barna, and Marin Litoiu. Economics-driven resource scalability on the cloud. In *11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS, page 129–139. ACM, 2016.

[21] Erik Fredericks. Automatically hardening a self-adaptive system against uncertainty. In *11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS. ACM, 2016.

[22] Erik Fredericks, Byron DeVries, and Betty Cheng. Towards run-time adaptation of test cases for self-adaptive systems in the face of uncertainty. In *9th Intl. Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS. ACM, 2014.

[23] Alessio Gambi, Daniel Moldovan, Georgiana Copil, Hong-Linh Truong, and Schahram Dustdar. On estimating actuation delays in elastic computing systems. In *8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS. IEEE, 2013.

[24] David Garlan, S-W Cheng, A-C Huang, Bradley Schmerl, and Peter Steenkiste. Rainbow: Architecture-based self-adaptation with reusable infrastructure. *Computer*, 37(10):46–54, 2004.

[25] Simos Gerasimou, Radu Calinescu, and Alec Banks. Efficient runtime quantitative verification using caching, lookahead, and nearly-optimal reconfiguration. In *9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS. ACM, 2014.

[26] Eoin M. Grua, Ivano Malavolta, and Patricia Lago. Self-adaptation in mobile apps: A systematic literature study. In *14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS, page 51–62. IEEE, 2019.

[27] Michele Guerriero, Damian Andrew Tamburri, and Elisabetta Di Nitto. Defining, enforcing and checking privacy policies in data-intensive applications. In *13th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS. ACM, 2018.

[28] Nikolas Roman Herbst, Samuel Kounev, Andreas Weber, and Henning Groenda. Bungee: An elasticity benchmark for self-adaptive iaas cloud environments. In *International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS. IEEE, 2015.

[29] Wei-Chih Huang and William J. Knottenbelt. Self-adaptive containers: Building resource-efficient applications with low programmer overhead. In *8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS, page 123–132. IEEE, 2013.

[30] Emilio Incerto, Mirco Tribastone, and Catia Trubiani. Symbolic performance adaptation. In *International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS. ACM, 2016.

[31] Pooyan Jamshidi, Aakash Ahmad, and Claus Pahl. Autonomic resource provisioning for cloud-based software. In *9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS, page 95–104. ACM, 2014.

[32] Pooyan Jamshidi, Javier Cámara, Bradley Schmerl, Christian Kästner, and David Garlan. Machine learning meets quantitative planning: Enabling self-adaptation in autonomous robots. In *14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS, page 39–50. IEEE, 2019.

[33] Pooyan Jamshidi, Miguel Velez, Christian Kästner, Norbert Siegmund, and Prasad Kawthekar. Transfer learning for improving model predictions in highly configurable software. In *12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS, page 31–41. IEEE, 2017.

[34] Jeffrey O Kephart and David M Chess. The vision of autonomic computing. *Computer*, (1):41–50, 2003.

[35] C. Kinneer, Z. Coker, J. Wang, D. Garlan, and C. Le Goues. Managing uncertainty in self-adaptive systems with plan reuse and stochastic search. In *International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE/ACM, May 2018.

[36] Jóakim V. Kistowski, Nikolas Herbst, Daniel Zoller, Samuel Kounev, and Andreas Hotho. Modeling and extracting load intensity profiles. In *10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS, page 109–119. IEEE, 2015.

[37] Jeff Kramer and Jeff Magee. Self-managed systems: An architectural challenge. In *Future of Software Engineering (FOSE)*, 2007.

[38] Christian Krupitzer, Felix Roth, Sebastian VanSyckel, Gregor Schiele, and Christian Becker. A survey on engineering approaches for self-adaptive systems. *Pervasive Mob. Comput.*, 17(PB):184–206, 2015.

[39] Satish Kumar, Tao Chen, Rami Bahsoon, and Rajkumar Buyya. Datesso: Self-adapting service composition with debt-aware two levels constraint

reasoning. In *15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS. ACM, 2020.

[40] Sara Mahdavi-Hezavehi, Vinicius H.S. Durelli, Danny Weyns, and Paris Avgeriou. A systematic literature review on methods that handle multiple quality attributes in architecture-based self-adaptive systems. *Information and Software Technology*, 90:1–26, 2017.

[41] Julie A. McCann and Markus C. Huebscher. Evaluation issues in autonomic computing. In Hai Jin, Yi Pan, Nong Xiao, and Jianhua Sun, editors, *Grid and Cooperative Computing - GCC 2004 Workshops*, pages 597–608. Springer, 2004.

[42] Jhonny Mertz and Ingrid Nunes. On the practical feasibility of software monitoring: A framework for low-impact execution tracing. In *14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS, page 169–180. IEEE, 2019.

[43] Gabriel A. Moreno, Alessandro V. Papadopoulos, Konstantinos Angelopoulos, Javier Cámara, and Bradley Schmerl. Comparing model-based predictive approaches to self-adaptation: Cobra and pla. In *12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS, page 42–53. IEEE, 2017.

[44] Henry Muccini, Mohammad Sharaf, and Danny Weyns. Self-adaptation for cyber-physical systems: A systematic literature review. In *11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS, page 75–81. ACM, 2016.

[45] Ali Yadavar Nikravesh, Samuel A. Ajila, and Chung-Horng Lung. Towards an autonomic auto-scaling prediction system for cloud resource provisioning. In *10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS. IEEE, 2015.

[46] Peyman Oreizy, Michael M. Gorlick, Richard N. Taylor, Dennis Heimbigner, Gregory Johnson, Nenad Medvidovic, Alex Quilici, David S. Rosenblum, and Alexander L. Wolf. An architecture-based approach to self-adaptive software. *IEEE Intelligent Systems*, 14(3):54–62, 1999.

[47] Ashutosh Pandey, Gabriel A. Moreno, Javier Cámara, and David Garlan. Hybrid planning for decision making in self-adaptive systems. In Giacomo Cabri, Gauthier Picard, and Niranjan Suri, editors, *10th IEEE International Conference on Self-Adaptive and Self-Organizing Systems, SASO 2016, Augsburg, Germany, September 12-16, 2016*, pages 130–139. IEEE Computer Society, 2016.

[48] Gustavo G. Pascual, Mónica Pinto, and Lidia Fuentes. Run-time adaptation of mobile applications using genetic algorithms. In *8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS, page 73–82. IEEE, 2013.

[49] Tharindu Patikirikorala, Alan Colman, Jun Han, and Liuping Wang. A systematic survey on the design of self-adaptive software systems using control engineering approaches. In *7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS, page 33–42. IEEE, 2012.

[50] Diego Perez-Palacin, Raffaela Mirandola, and José Merseguer. On the relationships between qos and software adaptability at the architectural level. *Journal of Systems and Software*, 87:1–17, 2014.

[51] Kai Petersen, Sairam Vakkalanka, and Ludwik Kuzniarz. Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, 64:1 – 18, 2015.

[52] João Felipe Pimentel, Leonardo Murta, Vanessa Braganholo, and Juliana Freire. A large-scale study about quality and reproducibility of jupyter notebooks. In *16th International Conference on Mining Software Repositories*, MSR '19, page 507–517. IEEE, 2019.

[53] Barry Porter, Roberto Filho, and Paul Dean. A survey of methodology in self-adaptive systems research. In *International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*, pages 168–177. IEEE, August 2020.

[54] Evangelos Pournaras, Mark Ballandies, Dinesh Acharya, Manish Thapa, and Ben-Elias Brandt. Prototyping self-managed interdependent networks: Self-healing synergies against cascading failures. ACM, 2018.

[55] Federico Quin, Danny Weyns, Thomas Bamelis, Sarpreet Singh Buttar, and Sam Michiels. Efficient analysis of large adaptation spaces in self-adaptive systems using machine learning. In *Software Engineering for Adaptive and Self-Managing Systems*, SEAMS. IEEE, 2019.

[56] Claudia Raibulet, Francesca Arcelli Fontana, Rafael Capilla, and Carlos Carrillo. An overview on quality evaluation of self-adaptive systems. In *Managing Trade-Offs in Adaptable Software Architectures*. 2017.

[57] Paul Ralph, Sebastian Baltes, Domenico Bianculli, Yvonne Dittrich, et al. ACM SIGSOFT Empirical Standards. https://arxiv.org/abs/2010.03525, 2020. Version 0.1.0, October 07, 2020.

[58] Oliviero Riganelli, Daniela Micucci, and Leonardo Mariani. Policy enforcement with proactive libraries. In *12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS, page 182–192. IEEE, 2017.

[59] Gordon S. Blair, Nelly Bencomo, and Robert France. Models@ run.time. *Computer*, 42:22 – 27, 11 2009.

[60] S. Shevtsov, D. Weyns, and M. Maggio. Handling new and changing requirements with guarantees in self-adaptive systems using simca. In *2017 IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 12–23, 2017.

[61] Stepan Shevtsov, Mihaly Berekmeri, Danny Weyns, and Martina Maggio. Control-theoretical software adaptation: A systematic literature review. *IEEE Trans. on Software Engineering*, 44(8):784–810, 2018.

[62] Seung Yeob Shin, Shiva Nejati, Mehrdad Sabetzadeh, Lionel C. Briand, Chetan Arora, and Frank Zimmer. Dynamic adaptation of software-defined networks for iot systems: A search-based approach. In *15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS, page 137–148. ACM, 2020.

[63] Bruno L. Sousa, Mívian M. Ferreira, Kecia A. M. Ferreira, and Mariza A. S. Bigonha. Software engineering evolution: The history told by icse. In *XXXIII Brazilian Symposium on Software Engineering*, SBES '19, page 17–21. ACM, 2019.

[64] Gustavo Sousa, Walter Rudametkin, and Laurence Duchien. Extending dynamic software product lines with temporal constraints. In *12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS, page 129–139. IEEE, 2017.

[65] Stefan Taranu and Jens Tiemann. On assessing self-adaptive systems. In *8th IEEE International Conference on Pervasive Computing and Communications Workshops*, PERCOM, pages 214–219, 2010.

[66] Thein Tun, Mu Yang, Arosha. K. Bandara, Yijun Yu, Armstrong Nhlabatsi, Niamul Khan, Khaled Khan, and Bashar Nuseibeh. Requirements and specifications for adaptive security: Concepts and analysis. In *Software Engineering for Adaptive and Self-Managing Systems*, 2018.

[67] Jeroen Van Der Donckt, Danny Weyns, Federico Quin, Jonas Van Der Donckt, and Sam Michiels. Applying deep learning to reduce large adaptation spaces of self-adaptive systems with multiple types of goals. In *Software Engineering for Adaptive and Self-Managing Systems*, SEAMS. ACM, 2020.

[68] Rini Van Solingen, Vic Basili, Gianluigi Caldiera, and H Dieter Rombach. Goal question metric (GQM) approach. *Encyclopedia of software engineering*, 2002.

[69] Norha M. Villegas, Hausi A. Müller, Gabriel Tamura, Laurence Duchien, and Rubby Casallas. A framework for evaluating quality-driven self-adaptive software systems. In *International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. ACM, 2011.

[70] Thomas Vogel and Holger Giese. Model-driven engineering of self-adaptive software with EUREMA. *ACM Transactions on Autonomous and Adaptive Systems*, 8(4), 2014.

[71] Maike Vollstedt and Sebastian Rezat. An introduction to grounded theory with a special focus on axial coding and the coding paradigm. In *Compendium for Early Career Researchers in Mathematics Education*, pages 81–100. Springer, 2019.

[72] Pascal Weisenburger, Manisha Luthra, Boris Koldehofe, and Guido Salvaneschi. Quality-aware runtime adaptation in complex event processing. In *12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS, page 140–151. IEEE, 2017.

[73] Danny Weyns. Software engineering of self-adaptive systems. In *Handbook of Software Engineering*, pages 399–443. Springer, 2019.

[74] Danny Weyns. *Introduction to Self-Adaptive Systems, A Contemporary Software Engineering Perspective*. Wiley, 2020.

[75] Danny Weyns and Tanvir Ahmad. Claims and evidence for architecture-based self-adaptation: a systematic literature review. In *European Conference on Software Architecture*, pages 249–265. Springer, 2013.

[76] Danny Weyns, M Usman Iftikhar, Sam Malek, and Jesper Andersson. Claims and supporting evidence for self-adaptive systems: A literature study. In *7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 89–98. IEEE, 2012.

[77] Claes Wohlin, Per Runeson, Martin Hst, Magnus C. Ohlsson, Bjrn Regnell, and Anders Wessln. *Experimentation in Software Engineering*. Springer, 2012.

[78] Valentina Zanardi and Licia Capra. Dynamic updating of online recommender systems via feed-forward controllers. In *6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS, page 11–19. ACM, 2011.

[79] Carmen Zannier, Grigori Melnik, and Frank Maurer. On the success of empirical studies in the international conference on software engineering. In *28th International Conference on Software Engineering*, ICSE, page 341–350. ACM, 2006.