Protocol for Mapping Study on Experimental Evaluations in Self-Adaptive Systems

A Ten-Years Perspective of SEAMS

Ilias Gerostathopoulos, Thomas Vogel, Danny Weyns, Patricia Lago

February 12, 2021

Abstract

With the increase of research in self-adaptive systems, there is a growing need to better understand the way research contributions are evaluated and how these evaluations are reported. Such insights will support researchers to better compare new findings, incrementally developing new knowledge for the community. However, so far there is no clear overview of the state of the art in how evaluations are performed in self-adaptive systems. To address this gap, we conduct a mapping study. The study focuses on experimental evaluations published at the prime venue of research in software engineering for self-adaptive systems in the last decade (2011-2020)—the International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS). Guided by a widely accepted process for conducting experiments, our literature review is centered on (i) the scope and goals of experiments reported by the SEAMS community, (ii) the way such experiments are designed and executed, and (iii) the way the results of such experiments are analyzed and packaged. This document provides the protocol of the study.

1 Motivation and Review Goal

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 become self-adaptive. Self-adaptation is typically realized via one or more 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 [28], initiated by the pioneering vision of IBM's autonomic computing [12] and the seminal work of Oreizy and colleagues [18].

Numerous novel approaches focusing on a variety of different aspects of engineering self-adaptive systems (runtime models, goal models, feedback controllers, modeling languages, verification at runtime, planning, etc.) have been proposed and evaluated by the research community over the past years. To that end, a number of testbeds, exemplars, and reusable artifacts have been developed and released 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 the state of the art related to both the contributions that have been proposed and the way these contributions are evaluated. While literature reviews have shed light on the contributions in the field [14, 19, 17, 16, 4, 9], the evaluation aspect has been less investigated. In particular, evaluations have been considered in self-adaptation reviews focusing on claims and evidence [30, 29] and methodology [21]. Yet, to the best of our knowledge, no study has targeted an in-depth analyze and characterization of the way experimental evaluations have been conducted and reported.

However, evaluation is central to self-adaptive systems (as for any other types of systems in software engineering), since contributions must be assessed on their added value and contribution [2]. Yet, evaluating contributions of self-adaptive systems may raise specific challenges due a high degree of automation of these systems and their ability to deal wit uncertainty during operation. Understanding the state of the art in conducting and reporting evaluations in self-adaptive systems can support researchers to better compare new findings to incrementally developing new knowledge for the community. Hence, it is important to provide an overview of evaluations of self-adaptive systems, which is currently missing.

To fill this gap, we perform a mapping study [20]. The goal of the study is to structure the evaluation of self-adaptive systems, i.e., providing an overview of the way evaluations are conducted in self-adaptive systems. We focus on experimental evaluations, i.e., evaluations that use one or more experiments, since performing experiments is the most common evaluation method used in self-adaptive systems. Concretely, the study is centered on (i) the scope and goals of experiments, (ii) the way experiments are designed and operated, and (iii) the way the results of such experiments are analyzed, packaged, and presented.

The remainder of this protocol is structured as follows. Section 2 provides the background and explains the focus of the mapping study. In Section 3, we define research questions, the searched sources, the inclusion and exclusion criteria, the data items that will be collected for papers along with the data-gathering procedure, and the approach we will use for analysis.

2 Background and Focus of the Mapping Study

2.1 Self-Adaptive Systems

This study focuses on what is commonly known as architecture-based adaptation [**?**, 7, 13, 32]. Architecture-based adaptation is a widely applied approach to realize self-adaptation, see [28] for an overview. Figure 1 shows the basic

[1] For an overview of exemplars published at the International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS) visit http://self-adaptive.org/exemplars.

decomposition of a self-adaptive system. 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.

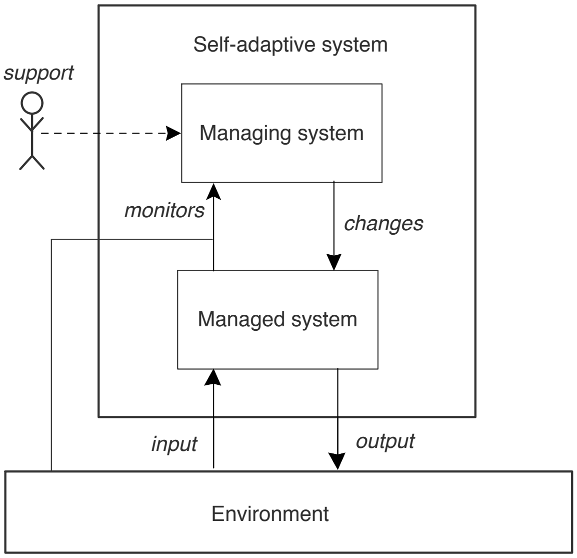

Figure 1: Basic decomposition of a self-adaptive system.

The managing system forms a feedback loop that comprises four essential functions: Monitor-Analyze-Plan-Execute that share Knowledge [12], MAPE or MAPE-K in short. The monitor tracks the managed system and the environment in which the system operates and updates the knowledge. The analyzer uses the up-to-date knowledge to evaluate the need for adaptation. If adaptation is required, it analyses alternative configurations of the managed system. We refer to these alternative configurations as the adaptation options. The planner then selects the best option based on the adaptation goals and generates a plan to adapt the system from its current configuration to the new configuration. Finally, the executor executes the adaptation actions of the plan. It is important to highlight that MAPE provides a reference model that describes a managing system's essential functions and the interactions between them. A concrete architecture maps the functions to corresponding components, which can be a one-to-one mapping or any other mapping, such as a mapping of the analysis and planning functions to one integrated decision-making component.

In this literature review, we consider papers that are based on the MAPE reference model that maps the MAPE functions (or some of them) to a specific component-based architecture.

2.2 Focus of Study

The overall goal of this mapping study is to understand how novel proposed engineering approaches for self-adaptation are evaluated in the International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS) papers. In order to get the focus of our review sharp, we performed a preliminary analysis and looked at the evaluation methods that were applied in full papers of SEAMS published from 2011 to 2020. After several iterations, we labelled the evaluation methods according to the following categories: no evaluation, survey, case study, proof, showcase, and experiment; see Table 1. Note that a single paper may use several methods or instances of the same method.

In our preliminary analysis, we found that more than 75% of the examined papers contained at least one experiment. With experiment, we do not necessarily mean controlled experiment as in [24], but a study with well-defined objectives that systematically evaluates a new solution providing quantitative results for more than one treatment. The other methods were far less used, with showcase being the second most used method. Finally, we found only a handful of papers containing a survey, case study[2], or proof.

Since experiments are more frequently used and more data points could be extracted from experimental evaluations to elicit significant reference guidelines, we decided to focus in our study on the evaluation method of *experiments*.

2.3 Basic Concepts of Experiments

In this section, we explain the basic concepts that we use in the study design. These concepts are based on the process and basic artifacts used in controlled experiments. 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. In particular, we focus on technology-oriented experiments that have systems and software elements as subject of the study (in contrast to studies with humans).

According to Wholin et al. [33], a controlled experiment is a well-defined empirical method that can be used to evaluate an idea or belief of a cause and effect relationship between constructs. More specifically, a researcher may have a theory or formulate a hypothesis that formalizes the idea or belief[4], and an experiment is used to test the theory or hypothesis.

For this purpose, an experiment studies the effect of manipulating one or more independent variables (i.e., factors) of the studied setting. The other independent variables are kept constant during the experiment so that the effect

[2] We mean here a case study as an empirical enquiry as for instance in `https://iansommerville.com/software-engineering-book/case-studies/` – the term is often not correctly used, which is a recurring issue also in other fields, see for instance [6].

[3] We consider a treatment to be a combination of factor values, contrary to Wohlin et al. who consider a treatment as "one particular value of a factor" [33, p. 75].

[4] Often researchers formulate research questions rather than formal hypotheses to capture the idea or belief of a cause and effect relationship between constructs.

Table 1: Evaluation methods with descriptions and examples. The number of papers with at least one occurrence of a certain method is shown in brackets after the method's name.

Evaluation method	Description
No evaluation (5)	The approach is not evaluated (i.e., no quantitative or qualitative evidence is provided). A concrete example may still be used, however, only for illustration and/or motivation of the work. An example is [15].
Survey (1)	An interview or a questionnaire has been used in the evaluation. A survey is "a system for collecting information from or about people to describe, compare or explain their knowledge, attitudes and behavior" [33, p. 10] and in our study takes the form of collecting data either through interviews or questionnaires. An example is [1].
Case study (0)	A case study has been used in the evaluation. A case study draws on multiple sources of evidence to investigate one instance of a phenomenon within its real-life context [33, p. 10]. We did not find an example of a SEAMS paper, but [11] is another example.
Proof (2)	A formal proof has been used in the evaluation. A proof employs mathematical reasoning to show that stated assumptions logically guarantee a conclusion (e.g. a theorem). An example is [3].
Showcase (27)	The evaluation presents results from a single treatment.[3] Such results can be quantitative or qualitative and can stem e.g. from one or more runs of a prototype realization of an approach or from a concrete application of an approach. An example is [10].
Experiment (66)	An experiment (Section 2.3) has been used in the evaluation. For an evaluation to qualify as experiment, *quantitative* results *for more than one* treatment need to be collected and presented. An example is [5]. A controlled experiment is an experiment that follows a rigorous well-defined process [33]. An example is [31].

on dependent variables caused by the manipulation of the factors can be measured [33]. Thus, an experiment considers multiple treatments (i.e., values of a factor), which allows researchers to compare the outcomes of the different treatments. This essentially tests the relationship between the treatment and the outcome and allows researchers to draw conclusions about the cause and effect relationship to which the theory or hypothesis refers.

To support researchers in setting up and conducting a successful experiment in software engineering, Wholin et al. [33] describe a process for experiments as shown in Figure 2. This process captures several aspects that are all considered relevant and important for experiments. The process starts with an *idea* that an experiment would be a reasonable option for an evaluation, for instance, evaluate a new analysis technique, and then comprises five steps:

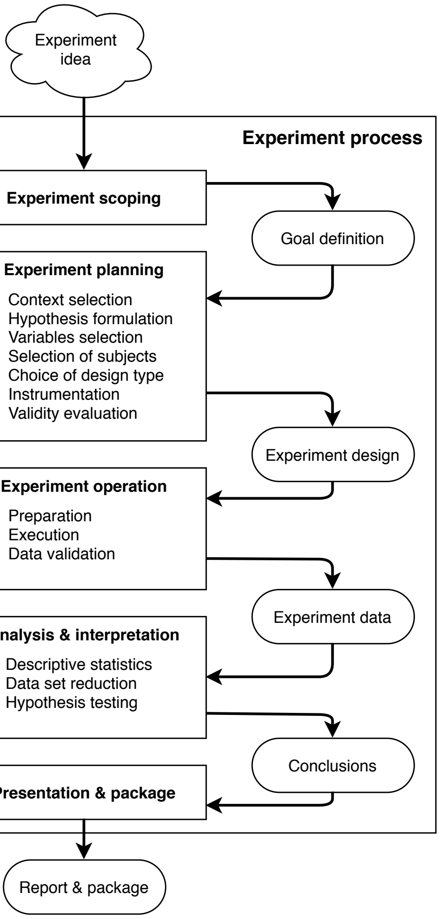

Figure 2: Experiments process and artifacts [33].

1. *Experiment scoping:* To scope an experiment, the goals of the experiment are defined. The goal definition comprises the object of the study, the purpose of the experiment, the effect under study, the perspective from which the results are interpreted, and the context in which the experiment is performed.

2. *Experiment planning:* The planning step refines the goal definition by de-

termining the experiment design that can be seen as the foundation of an experiment. The design requires from researchers to (i) select a context in which the experiment is carried out, (ii) formulate the hypothesis to be tested by the experiment, (iii) select the independent and dependent variables including the values they can take, (iv) select the subjects, (v) choose a design type (especially the factors and their individual treatments), (vi) determine the instrumentation by defining how the experiment should be executed and monitored, and (vii) evaluate the validity of the results and how threats could be mitigated by the experiment design.

3. *Experiment operation:* According to the experiment design, the experiment is prepared and executed. Additionally, the validity of the data collected during the experiment should be checked with respect to the execution and design of the experiment.

4. *Analysis & interpretation:* In this step, the data collected from the experiment is analyzed usually with descriptive statistics, then potentially reduced (e.g., by removing outliers), and finally used for testing the hypothesis with statistical tests. Then, the results of the analysis are interpreted to decide whether the hypothesis could be accepted or rejected and to draw conclusions from the experiment based on this decision.

5. *Presentation & package:* Finally, the results of the experiment are presented (e.g., in a report or research paper), packaged and made available for evaluation and to support replication.

3 Protocol Parts

This study uses the methodology of a mapping study, as described in [20]. The methodology defines the way in which a mapping study should be performed so that the relevant papers are properly identified, evaluated, and a map is produced (a map displays extracted knowledge, typically in some visual way, such as using flow charts, graphs, and Venn diagrams). The mapping study is composed of three stages: planning, execution, and reporting. During the planning stage, we define a protocol for the study. This protocol includes the research questions of the study, the sources to search for papers, the search string to collect papers, inclusion and exclusion criteria to select relevant papers, and the data items that need to be collected from the selected papers to answer the research questions. In the execution phase the search of the papers is applied and data is collected. In the reporting phase we organize the collected data and answer the research questions, document useful insights, and discuss potential threats to the validity of the study.

We conduct the mapping study with four researchers that jointly developed the protocol. To ensure the validity of the protocol, it will be reviewed by researchers with expertise in self-adaptation and experts in experimental software engineering. Since we collect all papers from SEAMS 2011 to 2020, we do not perform a search (see Section 3.2 for details). Two researchers filter the collected papers according to the inclusion and exclusion criteria (see Section 3.3) and extract the data from the included papers (see Section 3.5). To avoid bias, these two researchers crosscheck each others' results. If conflicts occur, the two researchers discuss the corresponding papers to resolve the conflicts. If a conflict cannot be resolved by the two researchers, the other two researchers are involved in the discussion to reach a consensus. Finally, all of the four researchers process the data, answer the research questions, and write a report.

3.1 Research Questions

We formulate the goal of the study using the classic Goal-Question-Metric (GQM) approach [26]:

> *Purpose*: Organize and characterize
> *Issue*: the way experimental evaluations are performed
> *Object*: in research on self-adaptation published at recent SEAMS installments
> *Viewpoint*: from a researcher's viewpoint.

We translate the overall goal of the study in five concrete research questions that correspond to the five phases of the experiment process proposed by Wohlin et al. [33] (see Figure 2):

RQ1: What is the scope of experiments?

With RQ1, we want to characterize the scopes of the experiments. This will help us better understanding the evaluated contributions, the purpose and object of evaluations, and identify possible correlations between these properties and other evaluation aspects.

RQ2: What is the experimental design of experiments?

With RQ2, we want to get a view on the design of experiments and identify characteristics specific to self-adaptive systems. We are interested in an in-depth description of independent and dependent variables, treatments, and designs (e.g. full factorial, partial factorial). This will shed light on the complexity and variability of experiments and present the different options a self-adaptive researcher has when designing their experiments.

RQ3: How are experiments operated?

With RQ3, we want to characterize the operation of experiments, such as how self-adaptive system-specific aspects are handled (e.g., the managed system and its execution). This will provide an overview of the different experiment environments and testbeds used so far.

RQ4: How is the experiment data analyzed?

With RQ4, we want to get insights of how experiment results are analyzed (e.g., using descriptive or inferential statistics).

RQ5: How are the results of experiments packaged?

With RQ5, we want to obtain an overview of how experiment results are packaged (e.g., in replication packages).

3.2 Searched Strategy

In our study, we examine the studies published at the main venue on engineering self-adaptive systems, i.e., the International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), covering the topic of experimental evaluations. To do so, we do not need to use a search string since we can easily identify all papers published in the last ten years of SEAMS (2011-2020). We do not use any expected technique to mitigate publication and sampling bias (e.g. manual and keyword automated searches, backward and forward snowballing searches, checking profiles of prolific authors in the area). According to the ACM SIGSOFT Empirical Standards [22], which is currently under development, this is an acceptable deviation to the way literature studies should be performed when focusing on one specific venue.

Other important self-adaptation venues include the IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS) and its precursors[5] and ACM Transactions on Autonomous and Adaptive Systems (TAAS). Yet, we focus our study on SEAMS for two reasons. First, there is a normative justification. Studies presented at SEAMS provide a representative sample of software engineering research of self-adaptive systems. Other studies have also chosen to focus on proceedings of specific venues, examples are ICSE [35], WICSA [6], and SBES [25]. The ACM SIGSOFT Empirical Standards considers this also as an acceptable deviation of the general principles of systematic literature studies [22]. Second, there is a qualitative justification. To make a useful and accurate assessment of the features we target in this review (the detailed data items are presented below), 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 some degree of bias of the focus on SEAMS. We will take this into account as a threat to validity.

3.3 Inclusion and Exclusion Criteria

We use the following inclusion criterion to select papers:

- **IC1: The paper is published at SEAMS between 2011 and 2020 (inclusive).**

  We selected the period from 2011 to 2020 since SEAMS became a symposium in 2011 (before it had been an ICSE workshop). Transforming

[5] ACSOS emerged in 2020 as a merger of the IEEE International Conference on Autonomic Computing (ICAC) and the IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO).

from a workshop to a symposium, we reckon that the rigor of the work published at SEAMS starting with 2011 has increased significantly, which reflects in an enhanced maturity level of the evaluations.

- **IC2: The paper empirically evaluates an approach using one or more experiments.**

  As mentioned in Sections 2.2 and 2.3, we focus on technology-centered experiments as it is the most common evaluation method used in SEAMS papers. As an example, by applying IC2, the paper [10] is excluded since it contains only a showcase and not an experiment, and the paper [5] is included as it includes an experiment.

We use the following exclusion criteria:

- **EC1: The paper is not a full research paper.**

  The motivation of EC1 is to exclude certain types of papers published at SEAMS, whose goal is typically not to propose and empirically evaluate an approach but rather presenting preliminary work (short papers), experience reports of applying self-adaptation in practice (experience papers), artifacts (artifact papers), tools (tool demonstration papers), Ph.D. projects (doctoral symposium papers), summaries of talks (keynote abstracts), or positions in a panel or debate (community debate paper). Although papers of these types may present parts of an experiment, they do not provide sufficient data to draw reliable or complete knowledge.

- **EC2: The paper presents a secondary study (e.g., literature review, survey, or mapping study) or an overview of the field (e.g., taxonomy, roadmap).**

  EC2 is motivated by our interest in primary studies that introduce (novel) approaches that are empirically evaluated using experiments, and not in secondary studies, such as literature reviews or surveys. Similarly, we are not interested in papers providing overviews of the field or roadmaps, as such work does not present and evaluate an approach. As an example, papers [8] and [34] are excluded as surveys and paper [23] as taxonomy.

Given these criteria, a paper is selected if it meets all of the inclusion criteria and does not meet any exclusion criterion.

3.4 Methodology

In conducting this study, we follow the six steps shown in the process of Figure 3. First, we create the initial set of primary studies by collecting all the papers published at SEAMS within 2011-2020. Second, based on the metadata associated with each paper, we exclude all papers that are not full research papers. Third, we examine all papers by first going through the title and abstract and, when needed, then the rest of the paper to identify and filter out cases

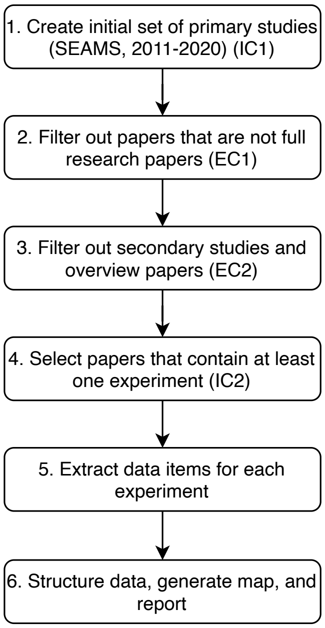
1. Create initial set of primary studies (SEAMS, 2011-2020) (IC1)
2. Filter out papers that are not full research papers (EC1)
3. Filter out secondary studies and overview papers (EC2)
4. Select papers that contain at least one experiment (IC2)
5. Extract data items for each experiment
6. Structure data, generate map, and report

Figure 3: Methodology of this study.

that are either secondary studies or overview papers. Fourth, we examine the remaining papers, by scanning the whole papers to select papers that contain at least one experiment. We read the remaining papers in full, focusing on the evaluation sections, identify all the experiments contained in each paper, and extract data items for each identified experiments. Finally, we structure the collected data, generate the map, and write the final report of the study.

3.5 Data Items

To answer the research questions and create a map, we define a set of data items to be extracted from the papers. Table 2 gives an overview of the data items. We briefly describe each data item and present the concrete options for each data item.

Since the data items refer to a single experiment and a paper may contain more than one experiment, as a first step, we extract all the experiments that are included in a paper and then extract data of each experiment independently.

F1 The target of evaluation: the main element that is subject of evaluation, incl. the whole feedback loop and methods for distinct MAPE-K stages and learning. Options are collected during data-gathering. Examples may include: planning method, monitoring method, analysis technique, learning approach.

Table 2: Data extraction items

ID	Item	Use
F1	Target of evaluation	RQ1
F2	Objectives of evaluation	RQ1
F3	Explicit formulation of evaluation problem	RQ2
F4	Constant independent variables	RQ2
F5	Blocking factors	RQ2
F6	Factors	RQ2
F7	Dependent variables	RQ2
F8	Counts of experiment variables	RQ2
F9	Design type	RQ2
F10	Managed system	RQ3
F11	Nature of managed system	RQ3
F12	Data provenance	RQ3
F13	Uncertainty	RQ3
F14	Type of analysis	RQ4
F15	Explicit answer to evaluation problem	RQ4
F16	Threats to validity/limitations	RQ4
F17	Results available	RQ5
F18	Replication package available	RQ5

F2 The aspects of the proposed approach that are evaluated. These are typically part of the overall goal of the experiment, mentioned explicitly or implicitly. Options are collected during data-gathering. Examples may include: effectiveness, efficiency, performance, cost, scalability, robustness.

F3 Captures whether there is an explicit formulation of the evaluation problem by either research questions or hypotheses. Options: Research questions, hypotheses, None.

F4 The name of the variables that remain constant across different experiment treatments. Options are collected during data-gathering. Examples may include: "system load" and "network configuration."

F5 The name and number of values of the the variables that are used to create experiment blocks. A blocking factor is an independent variable that probably has an effect on the response, but we are not interested in that effect [33, p. 94]. Hence, the effects between blocks are not studied. Options are collected during data-gathering. We use as a notation to refer to these variables as: “name (number of values).” Examples may include: “model complexity (3)”.

F6 The name and number of values of the variables that change across different experiment treatments. Options are collected during data-gathering. We use as a notation to refer to these variables as: “name (number of values).” Examples may include: “controllers pole (2)”.

F7 The name of the variables that measure the effect of a treatment (also called *response variables* [33, p. 74]). Options are collected during data-gathering. Examples may include: consumed energy, response time, requests per second, number of servers, planning time.

F8 The number of factor values that are actually used in the experiment. The factor names come from F6 and F5. The notation “name1 (number1 of values) x ... x nameN (numberN of values)” will be used. Examples may include: “scenario (2) x QoS modeling approach (2)”.

F9 The design type used in the experiment, following the standard design types of Wohlin et al. [33, p. 95]. This can be derived from F8, but is added to speed up the analysis process. Options: One factor with two values, One factor with more than two values, Two factors with two values, More than two factors each with two values, Other.

F10 The name of the managed system, if any. Initial options are the SEAMS artifacts that that can be used as managed systems (non-complete list available at http://self-adaptive.org): Hogna, TAS, DeltaIoT, UNDERSEA, CrowdNav, SAVE, mRUBIS, SWIM, DragonFly, DingNet, DARTSim. Additional options are collected during data-gathering.

F11 The type of managed system used in the evaluation. Options are: Model (e.g. the managed system is represented as a queuing model), Simulated/Emulated (e.g. a Cloud simulator or device emulator of the managed system), Real (e.g. a Java application or a physical deployment).

F12 Source of data related to the users or the environment of the managed system. Options: Synthetic data (e.g. made-up pattern of user requests), Emulated data (e.g. pattern of user requests created based on real-world data), Real-world data (pattern of user requests captured from real-world).

F13 The way uncertainty is represented in the experiment. This type of uncertainty can create the need for self-adaptation. Options are collected during data-gathering. An example is: the value of sensor measurements are taken from a particular probability distribution during a run.

F14 The type of analysis that is performed on the results of the experiment. Only the most prominent case is captured out of the following four options (with increasing prominence): None, Exposition (qualitative interpretation, narrative), Descriptive statistics (e.g. plots), Statistical tests.

F15 Captures whether there is an explicit answer to evaluation problem specified either as research questions or hypotheses. Options: yes, no, N/A (to be used if the answer to F3 is "none").

F16 The types of threats to validity mentioned (in a dedicated section/subsection or paragraph), if any. Options are one or more (when applicable) of: no, yes (without categories), internal, external, construct, conclusion.

F17 Whether the evaluation results are available (e.g. via a URL). Options: yes, no.

F18 Whether a full replication package (containing not only the implementation of the managed system, but also scripts to run experiments and data used as input – e.g. to perform a simulation or to parametrize a framework) is present (e.g. via a URL). Options: yes, no.

Figure 4 maps the research questions and data items of our study to the experiment process proposed by Wholin et al. [33] (cf. Figure 2). In our study we cover all of the aspects of the process that are considered as relevant and important for experiments by Wholin et al. [33] except the following ones (see gray aspects in Figure 4):

- *Selection of subjects* because based on our experience of the SEAMS venue we know that most experiments reported at SEAMS are technology-oriented. In such cases, "different technical treatments are applied to different objects" whereas in "human-oriented experiments, humans apply different treatments to objects" [33, p. 11]. Consequently, the subjects in technology-oriented experiments do not play a role and thus, we do not investigate the selection of subjects.

- *Instrumentation* because technology-oriented experiments (as reported in SEAMS) are concerned with measurements performed in software systems at a technical level. Thus, the technological instrumentation can be specific to each individual system and we would not gain insights about the design of experiments by investigating the technologies used for instrumentation. However, we capture the dependent variables of the experiment that are eventually mapped to technological instruments for measurement.

- *Data validation* and *data set reduction*: we exclude validation of data and data set reduction from our study because they are specific aspects of experiments out of scope of what we target in this literature review.

Thus, our study covers the whole experiment process and key aspects that are relevant for technology-oriented experiments reported at SEAMS. This coverage gives us confidence that our study addresses all relevant aspects of experiments and therefore, will provide a comprehensive overview of SEAMS experiments.

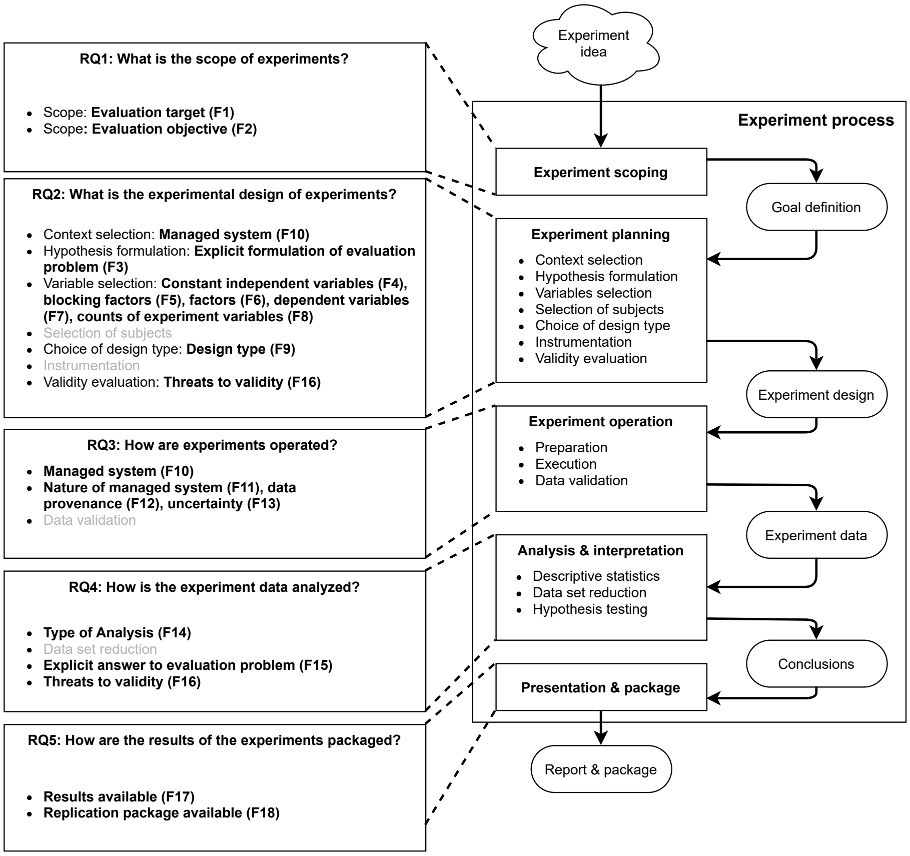
RQ1: What is the scope of experiments?
Scope: Evaluation target (F1)
Scope: Evaluation objective (F2)
RQ2: What is the experimental design of experiments?
Context selection: Managed system (F10)
Hypothesis formulation: Explicit formulation of evaluation problem (F3)
Variable selection: Constant independent variables (F4), blocking factors (F5), factors (F6), dependent variables (F7), counts of experiment variables (F8)
Selection of subjects
Choice of design type: Design type (F9)
Instrumentation
Validity evaluation: Threats to validity (F16)
RQ3: How are experiments operated?
Managed system (F10)
Nature of managed system (F11), data provenance (F12), uncertainty (F13)
Data validation
RQ4: How is the experiment data analyzed?
Type of Analysis (F14)
Data set reduction
Explicit answer to evaluation problem (F15)
Threats to validity (F16)
RQ5: How are the results of the experiments packaged?
Results available (F17)
Replication package available (F18)
Experiment idea
Experiment process
Experiment scoping
Goal definition
Experiment planning
Context selection
Hypothesis formulation
Variables selection
Selection of subjects
Choice of design type
Instrumentation
Validity evaluation
Experiment design
Experiment operation
Preparation
Execution
Data validation
Experiment data
Analysis & interpretation
Descriptive statistics
Data set reduction
Hypothesis testing
Conclusions
Presentation & package
Report & package

Figure 4: Research questions and data items mapped to the experiment process by Wholin et al. [33].

3.6 Approach for the Analysis

We tabulate the data in spreadsheets for processing. We use descriptive statistics to present and structure the quantitative aspects of the extracted data and summarize the data in a comprehensible format 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, F5, F6, F7, and F13 we collect free text and apply coding [27] to capture the essence of the answers to these items.

In the following subsections, we provide the labels derived for each of the above data items, for reproducibility's sake.

3.6.1 Target of Evaluation (F1)

Based on the MAPE-K reference model [12], we label the evaluation target to stages of the feedback loop (monitoring, analysis, planning, execution), to the whole feedback loops, and due to the recent trend of using machine learning to learning methods.

- *Feedback Loop Approach*
- *Monitoring Method*
- *Analysis Method*
- *Planning Method*
- *Execution Method*
- *Learning Method*

3.6.2 Evaluation Objectives (F2)

We grouped the evaluation objectives that we found in the primary studies in the following categories.

- *Effectiveness*: This refers to whether the adaptation goals are met. Either effectiveness was explicitly mentioned as the evaluation objective, or was derived when not stated explicitly and the evaluation's purpose was to show that an approach is either feasible or better than another. Also, under effectiveness fall the sub-objectives of quality improvement, and functional abilities (e.g. "ability to detect violations of real-time properties", "ability to reduce adaptation space").
- *Learning ability*: This refers to objectives that evaluate the capabilities of learning approaches with respect to learning-specific criteria such as accuracy, sensitivity, or correlation.
- *Time efficiency*: This refers to time or performance related experiments.
- *Scalability*: This refers to how a quality behaves when the an aspect of the system (e.g. size, resources) is increased or decreased.
- *Robustness*: This refers to experiments aiming at showing that the approach or system is not sensitive to different changes.
- *Other*: This includes less popular objectives such as benchmarking capabilities, fault detection capabilities, resource utilization, and SASO properties evaluation.

3.6.3 Constant Independent Variables, Blocking Factors, and Factors (F4, F5, F6)

The independent variables are labeled with the basic elements of a self-adaptive system (see Section 2.1), to which they refer.

- *Managing system*: Here we distinguished between *Method* and *Parameter* based on whether the variable relates to a whole managing system method or a parameter of it, respectively.
- *Managed system*: Here we distinguished between *Different* and *Variation* based on whether the variable relates to a whole managed system or a variation of it, respectively.
- *Environment*
- *Goals*

If a variable refers to more than one of these elements, we use the label *Cross-cutting*.

3.6.4 Dependent variables (F7)

To label the dependent variables, we use the quality characteristics defined in ISO/IEC 25010 (see Table 3)[6]. This standard provides a comprehensive list of quality aspects that refer to stakeholder's needs, which can be used to evaluate a system in terms of meeting those needs.

3.6.5 Uncertainty (F13)

For each uncertainty item, we provided a label for both its type and its representation.

We used the following as labels for the type of uncertainty:

- *Human*
- *Context*
- *System*
- *Goals*

We used the following as labels for the representation of uncertainty:

- *Random*
- *Deterministic* (Called "Predefined" in the paper)
- *Probabilistic*

If the representation of uncertainty is not clearly described or cannot be derived by interpretation, we label the representation as *Unclear*.

[6] https://iso25000.com/index.php/en/iso-25000-standards/iso-25010

Table 3: Quality characteristics of the ISO/IEC 25010 product quality model.

Quality characteristic	Sub-characteristics
Functional Suitability	Functional completeness Functional correctness Functional appropriateness
Performance efficiency	Time behaviour Resource utilization Capacity
Compatibility	Co-existence Interoperability
Usability	Appropriateness recognizability Learnability Operability User error protection User interface aesthetics Accessibility
Reliability	Maturity Availability Fault tolerance Recoverability
Security	Confidentiality Integrity Non-repudiation Accountability Authenticity
Maintainability	Modularity Reusability Analysability Modifiability Testability
Portability	Adaptability Installability Replaceability

References

[1] A. Bennaceur, A. Zisman, C. McCormick, D. Barthaud, and B. Nuseibeh. Won't take no for an answer: Resource-driven requirements adaptation. In *2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 77–88, 2019.

[2] 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.

[3] Paulo Casanova, David Garlan, Bradley Schmerl, and Rui Abreu. Diagnosing unobserved components in self-adaptive systems. In *9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '14, page 75–84. ACM, 2014.

[4] 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 '19, page 13–24. IEEE, 2019.

[5] 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 '16, page 129–139. ACM, 2016.

[6] M. Galster and D. Weyns. Empirical research in software architecture: How far have we come? In *2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, pages 11–20, 2016.

[7] 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.

[8] E. M. Grua, I. Malavolta, and P. Lago. Self-adaptation in mobile apps: a systematic literature study. In *2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 51–62, 2019.

[9] 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 '19, page 51–62. IEEE, 2019.

[10] M. Usman Iftikhar and Danny Weyns. Activforms: Active formal models for self-adaptation. In *Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS

’14, page 125–134, New York, NY, USA, 2014. Association for Computing Machinery.

[11] Didac Gil De La Iglesia and Danny Weyns. Mape-k formal templates to rigorously design behaviors for self-adaptive systems. *ACM Trans. Auton. Adapt. Syst.*, 10(3), September 2015.

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

[13] Jeff Kramer and Jeff Magee. Self-managed systems: an architectural challenge. In *Future of Software Engineering (FOSE ’07)*, pages 259–268, 2007.

[14] 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.

[15] V. P. La Manna, J. Greenyer, C. Ghezzi, and C. Brenner. Formalizing correctness criteria of dynamic updates derived from specification changes. In *2013 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 63–72, 2013.

[16] 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.

[17] 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 ’16, page 75–81. ACM, 2016.

[18] 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.

[19] 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 ’12, page 33–42. IEEE, 2012.

[20] 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.

[21] 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, aug 2020.

[22] 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.

[23] A. J. Ramirez, A. C. Jensen, and B. H. C. Cheng. A taxonomy of uncertainty for dynamically adaptive systems. In *2012 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 99–108, 2012.

[24] D. I. K. Sjoeberg, J. E. Hannay, O. Hansen, V. B. Kampenes, A. Karahasanovic, N. . Liborg, and A. C. Rekdal. A survey of controlled experiments in software engineering. *IEEE Transactions on Software Engineering*, 31(9):733–753, 2005.

[25] 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.

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

[27] 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.

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

[29] 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.

[30] 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.

[31] Danny Weyns, M. Usman Iftikhar, and Joakim Söderlund. Do external feedback loops improve the design of self-adaptive systems? a controlled experiment. In *Proceedings of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '13, page 3–12. IEEE Press, 2013.

[32] Danny Weyns, Sam Malek, and Jesper Andersson. Forms: Unifying reference model for formal specification of distributed self-adaptive systems. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 7(1):8, 2012.

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

[34] Eric Yuan and Sam Malek. A taxonomy and survey of self-protecting software systems. In *Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '12, page 109–118. IEEE Press, 2012.

[35] 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 '06, page 341–350. ACM, 2006.