A Framework for Prototyping and Evaluating Self-adaptive Systems - A Research Preview

Fabian Kneer and Erik Kamsties

Dortmund University of Applied Sciences and Arts,
Emil-Figge-Str. 42, 44227 Dortmund, Germany
{fabian.kneer,
erik.kamsties}@fh-dortmund.de
http://www.fh-dortmund.de/

Abstract. [**Context and motivation**] In the last years, a considerable number of solutions were developed for self-adaptive systems (SAS). These solutions are based on a feedback loop (MAPE loop), but employ different approaches to realize the different facets of the feedback loop. For example, the models used for the *analysis* step differ (e.g., goal-models, feature-models). [**Question/problem**] In order to build a self-adaptive system for a particular application, a researcher or practitioner has not only to implement the application itself, but also the feedback loop. That is, one has first to select an appropriate solution from the literature and second to implement and evaluate the solution in the environment at hand. [**Principal ideas/results**] The goal of our work is to reduce the effort for developing a SAS by providing a framework for prototyping and evaluation. This framework comprises (1) a set of open source implementations of selected approaches from which an engineer can choose, (2) a case study to execute the implementations, and (3) a set of metrics for data collection during execution. [**Contribution**] The engineer can use the framework as a sandbox to compare several approaches in order to make a more informed decision which approach to choose. Possibly, the prototypical implementation can also be used in the final SAS implementation. This is a research preview that reports first results about the implementation of selected approaches (1), setting up a case study (2), and it outlines concepts regarding evaluation (3).

Keywords: Self-adaptive systems, case studies, experimentation

1 Introduction

Following the ITU [10], the Internet of Things (IoT) is a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies. A *thing* is an object of the physical world (physical things) or the information world (virtual things), which is capable of being identified and integrated into communication networks. A *device* is a piece of equipment with the mandatory capabilities of communication and the

optional capabilities of sensing, actuation, data capture, data storage, and data processing. One key concern of systems operating in the IoT is to dynamically adapt to changing environments, due uncertainties during requirements-, design-, and run-time (see [12]).

A self-adaptive system (SAS) contains a feedback loop to react on uncertainties in the environment. This loop is called MAPE loop by IBM [7]: **M**onitor - **A**nalyze - **P**lan - **E**xecute. As a first step, a system has to monitor its environment and context with software and hardware agents like sensors. Then this information is filtered until an investigation-relevant phenomenon is detected. The system has to analyze these measured properties to check the effect on the system requirements to verify if a change is required. Next, a plan is generated which describes a reaction strategy on how to solve any potential issues. The last step is to execute the configuration strategy via actuators of the system.

A considerable number of concepts for self-adaptive systems has been developed. From a practitioner's perspective, open implementations of the MAPE feedback loop for prototyping purposes are missing. This is a show-stopper in practice, as a practitioner would need to work through research papers in order to build such a prototype. A researcher who is interested in the comparison, extension and/or application of existing solutions to a new domain is in a similar situation.

In this research preview we suggest a *prototyping and evaluation framework* for self-adaptive systems. The first goal of the framework is to ease the prototyping (and possibly development) of self-configuring systems. For this purpose, the framework offers implementations of selected approaches to SAS based on the MAPE loop (e.g., based on feature models as suggest by Pascual et al. [11]).

The second goal of the framework is to ease the evaluation of self-configuring systems, to allow for instance benchmarks between different approaches. For this purpose, the framework provides a case study drawn from the smart city domain, and a set of metrics for evaluation. The framework is able to collect data on a subset of the metrics at runtime about overall quality, effort, and cost.

The framework can be used in a number of ways on the given case study:

- to *understand* a particular SAS approach,
- to *optimize* the application of an approach, or
- to *compare* approaches in a particular target environment.

The following section reports on the related work regarding frameworks for construction and approaches to evaluation of self-adaptive systems. Section 3 describes the design of our framework for prototyping and evaluation, the current extent of implementation and first experiences. Section 4 outlines the next steps that we scheduled to complete the framework, and Section 5 concludes the paper by describing one envisioned application scenario in more detail.

2 Related Work

This section presents previous approaches to prototyping and evaluation of self-adaptive systems. Several implementations of the MAPE loop have been pub-

lished [5]. IBM's Autonomic Computing Toolkit[1] was introduced in 2004. The toolkit provides a practical framework and reference implementation for incorporating autonomic capabilities into software systems [5]. A so-called *autonomic manager* can be build with the toolkit, which is a prototype of the MAPE loop and can be used for analyzing logs of a managed application.

With respect to evaluation, metrics have been published. Some authors proposed metrics to compare SAS, for example McCann et al. [9]: (1a) Quality of Service, (b) Cost, (c) Granularity/Flexibility, (d) Failure avoidance (Robustness), (e) Degree of Autonomy, (f) Adaptivity, (g) Time to adapt and (h) Reaction Time, (i) Sensitivity and (j) Stabilization.

Gjorven et al. [4] view adaption as a *service* which results in the metric *Quality of Adaption (QoA)*. The authors defined the following properties that describe the QoA, (2a) safe adaption, (b) secure adaption, (c) optimal adaption, (d) cost of adaption, (e) performance of adaption and (f) human in the loop.

We surveyed the literature on metrics that are actually used to evaluate self-adaptive and self-configuring systems and identified the following metrics: (3) the number of needed reconfigurations vs optimal number of reconfigurations, (4) resource usage (a) power consumption, (b) memory size, (c) execution time of the optimization process or parts like initializations, (5) average satisfaction of elements over a time interval, (6) impact on the target system (a) average lines of code per module that must be change (b) average response time of the target system.

We observed that an approach is validated with an average of two metrics out of the list above. Furthermore, the different approaches vary in the chosen metrics. These facts make it difficult or even impossible to compare current approaches given the information provided in the publications.

Our framework follows previous work in that it supports to build a SAS. The difference lies in our goal also to evaluate and compare competing solutions.

3 Design and Implementation of the Framework

The framework bases on experiences we made with the implementation of a goal-oriented (i*-based) approach to self-adaptive systems [6]. We developed a generalized architecture for prototyping SAS in the spirit of a software product-line (SPL), by an analysis of *commonalities* and *differences* of approaches suggested for SAS. We implemented a feature-orientated approach (Pascual et al. [11]) to validate the initial architecture of the prototyping framework. The details of the design, implementation, and first experiences are described in the remainder of this section.

Design. Three important paradigms for self-configuring and adaptive systems can be distinguished:

– Goal Oriented Approaches e.g. Zanshin from Mylopoulos et al. [13]

[1] Autonomic Computing Toolkit Home page: `http://www.ibm.com/developerworks/autonomic/r3/overview.html`

– Dynamic Decision Networks by Bencomo et al. [2]
– Feature Models e.g. by Garcia-Galan et al. [3] or Pascual et al. [11]

The focus of these approaches mostly lies on a better reconfiguration of the system and also on a better decision about the need of a new configuration. This means to identify the best configuration for the current system situation.

The largest commonality is the architecture of the approaches to self-adaptive systems that we currently analyzed. All of them are based on a feedback loop (MAPE) that consists of the parts **m**onitor (monitoring configuration), **a**nalyze (on a Requirements model), **p**lan (decision algorithm & decision model or elements) and **e**xecute. The differences are in the way in which the different activities of the MAPE feedback cycle are carried out:

1. An *indicator* for an adaption can be Quality of Service for service-oriented systems or a variable (input, output, parameter) for embedded systems.
2. A monitor configuration can be a simple event-condition-action (ECA), an assertion, or a domain assumption. A monitor configuration describes a fact of the running systems that should be monitored because it can lead to a reconfiguration or adaption.
3. Different models are used for the *analyze* activity such as goal models or feature models. These models allow to represent requirements with alternative realization strategies.
4. The decision making is based on quality values for different realization strategies. For example in goal models, softgoals and contribution links are used to compute quality values, which in turn are used generate a reconfiguration plan.
5. The reconfiguration plan can be to increase or decrease values of variables or activate or deactivate components, modules, or functions.

The implementation of the framework presented in the following section is based on this analysis of commonalities and differences.

Implementation. The prototyping framework is implemented in the Python programming language as this language allows for a wide range of deployments and is frequently used for prototyping (yet not limited to that purpose). The framework contains components to implement the different activities of the MAPE loop as well as further functions that are needed to build and run an adaptive system.

Fig. 1 shows the concept of the prototyping framework. Fundamentally, we separate *development time* and *runtime* artifacts. The development time artifacts represent additional aspects to be captured in the requirements phase. Runtime artifacts are components required to build the SAS. To easy prototyping, some runtime artifacts are generated from the development time artifacts. This process is described in more detail in the subsection "Generator" on Page 6.

We start the detailed discussion of the framework with the development time artifacts. *Self-configuring Requirements* are requirements with changing indicator values and also a status label that shows if the requirements is active or

disabled for a system situation. *Indicators* are measurable elements that are gathered by hardware agents or software agents depending on the system. A *Monitor Configuration* describe how an indicator is measured and how an assured element change if the indicator change. A simple example is if the WLAN connection (indicator) of a system fails a realization strategy that needs this connection must be disabled. A *Requirements Model* contains the different realization strategies of a system. The model must be validated to verify that a configuration for a given situation is valid or an adaption has to be performed. The implementation of a *Validation Algorithm* differs because of the previous mentioned different requirements models. The last component of the framework is a *Decision Model*, which is used to find the best configuration for the current system situation. It contains all information to compute the value for a realization strategy. The computing is done by the *Decision Algorithm*. The result of the decision algorithm is a reconfiguration plan for the application.

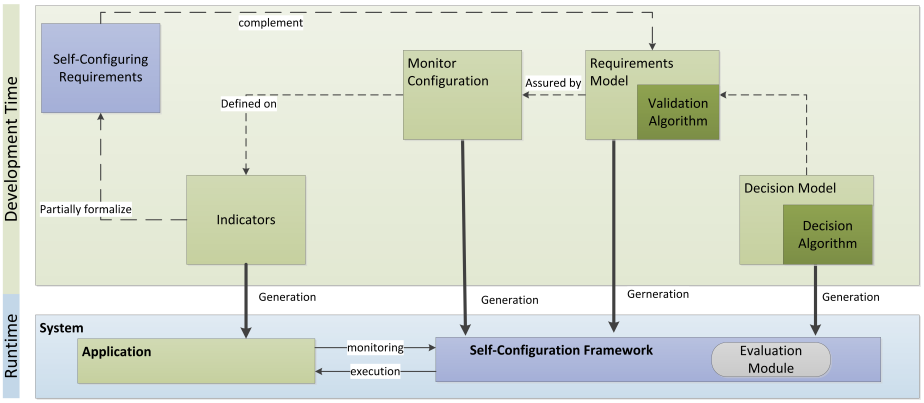

Fig. 1. Prototyping and Evaluation Framework

The prototyping framework provides a set of components from which can be chosen to develop a self-adaptive system. At the time of writing it contains a feature-oriented approach with utility functions to compute a configuration with a genetic algorithm (see Pascual et al. [11]).

The framework can be populated with further approaches. In order to give an impression of the required effort, we provide programming effort data of the feature-oriented approach mentioned above. The extension has taken about 3 man-week's and 1500 LoC:

- ECA's are used for monitoring the system (*400 LoC 3-4 Days*)
- Feature model is used as requirements model and representation of alternative realization strategies (*600 LoC 3-5 days*)
- Utility Table and - function are used to compute a configuration (*200 LoC 3 Days*)

– A genetic algorithm is used to choose the "best" configuration (*100 LoC 2 Days*)
– Configuration example (*150 LoC 1/2 day*)
– Simple Specification (1 textual requirement and feature model with 19 features) (*200 LoC 1/2 day*)

Generator. In order to ease the development of a prototype, a self-adaptive system is partially *generated*. That is, during development time (see upper part of Fig. 1), only the essential information for building the SAS is specified by the user. When using the feature-oriented approach this is: *indicators* (variables of the systems with a type), a *feature model* (different features of the systems and variation points), *ECA rules* (indicator change event, boolean condition, change action on a element, e.g. feature), *utility function* (table with utility values of the different variations of the feature model).

Probes for an application and the whole self-configuring framework (with integrated MAPE loop), see bottom part Fig. 1 can now be *generated* out of a specification. The indicators are used to generate probes for an application. These probes are used by the self-configuring framework to monitor the indicators and check the related ECA rules. The ECA description is used to generate the rules of the ECA rule engine. An application that include the generated probes can use the self-configuring systems to adapt the behavior to the changing context.

If the specification changes because new elements like indicators, ECA rules, features or utility elements are added or elements are refined for example an ECA rule change, only the generation process is re-triggered and the new prototype can be used for the system which contains the probes. Further details about the generator concept are given in [8].

4 Roadmap

The next steps on our roadmap to complete the framework are described in the following.

Extension of Framework. Our next step is to re-implement our goal-oriented approach to self-adaptive systems based on i* described in [6]. This will allow for comparisons which are not possible at the time of writing.

Case Study. Within the Internet of Things, we have identified *smart cities* as an interesting domain for a case study. It is accessible to many readers, it is complex, and comprises many different facets. A hardware demonstrator of a smart public lighting systems is under development that includes dynamic street lights which adapt there light color and light intensity to its environment and also communicate with other systems for example to display free parking-lots in navigation systems.

Evaluation. A component for logging and monitoring metrics is under development. Given the overall structure of our framework shown in Fig. 1, the data on the following metrics can be automatically collected:

- *Cost*: Some possible indicators are memory size, CPU or power consumption.
- *Failure Avoidance*: how much time elapse until a failure is fixed.
- *Human in the loop/ Degree of Autonomy* : count of the interaction with a user over a specified time interval.
- *Optimal Adaption*: collect the satisfaction of configurations and count of adaptations during a specified time interval. Build the average of these values and compare the values with a defined ideal state.
- *Adaption/ Reaction Time*: execution time of an adaption process.
- Safe Adaption: count of failures (e.g. real-time constrains of an embedded system) that are produced because of an adaption.
- *Stabilization*: that is the time taken from a change in the systems environment to stabilize the operations of a system.

The evaluation is limited to the metrics that can be captured at *runtime*.

5 Envisioned Usage of Framework

Under the assumption that a researcher has developed a SAS prototype (using feature models, ECAs, and a utility function as described before), we sketch in this section a scenario for evaluation.

Measurement is an established discipline in software engineering and there is a large consensus that measurement has to conducted in an goal-oriented fashion. The Goal-Question-Metric paradigm [1] is one approach to systematically derive metrics from measurement goals.

A goal for a researcher could be to analyze the *decision process of the MAPE loop* for the purpose to *understand* with the focus on *performance of the decision process* from the perspective of *a researcher* in the context of *an SAS implementation on a Raspberry Pi*. Following the GQM approach, questions are derived to capture quality factors and factors that are hypothesized to influence the quality factors. Questions are further refined into metrics. Table 1 shows a possible GQM measurement model for this evaluation.

6 Conclusion

In this research preview we outlined a prototyping and evaluation framework for self-adaptive systems (SAS) and sketched its application. The architecture of the framework is based on the MAPE loop and it will offer a number of components implementing different approaches to SAS.

One major benefit of the framework is to ease the development of self-adaptive systems by a code generator and components that can be used out

Goal	G1	Understand the performance of decision process.
Question	Q1	What is the current adaption and decision processing performance
Metrics	M1	Average execution time
	M2	Average execution time of the decision process
Question	Q2	What resources are currently used by the decision process related to the adaption process
Metrics	M3	Average memory size of $\frac{decision\ process}{adaption\ process} * 100$
	M4	Average power consumption of $\frac{decision\ process}{adaption\ process} * 100$
	M5	Average CPU load of $\frac{decision\ process}{adaption\ process} * 100$
Question	Q3	What is the current quality of the adaption process
Metrics	M6	Average utility
	M7	Average number of adaptations

Table 1. Example GQM table for Evaluation of the Decision Process

of the box. That is, the user of the framework is able to focus on the application itself and on the definition of the adaptation requirements in the selected formalism (e.g., feature models or goal models). The framework may also serve as a basis for 3rd party implementations.

The other major benefit is to provide a common basis for evaluation of approaches to SAS. For this purpose, a component is currently added the framework that automates data collection at execution time. The metrics that could be collected with the help of the framework were outlined.

As this paper is a research preview we have not paid much attention to validation. Yet, one major question to answer is how much effort the framework actually saves in prototyping a SAS.

Finally, we believe that commonly available, open-source implementations of self-adaptive systems help to promote the field. Therefore, it is planed to make the framework open source and publicly available using GitHub, when it has passed a first validation.

References

[1] V. R. Basili, G. Caldiera, and H. D. Rombach. "The Goal Question Metric Approach". In: *Encyclopedia of Software Engineering*. Wiley, 1994.

[2] N. Bencomo and A. Belaggoun. "Supporting Decision-Making for Self-Adaptive Systems: From Goal Models to Dynamic Decision Networks". In: *Requirements Engineering: Foundation for Software Quality*. Ed. by J. Doerr and A. Opdahl. Vol. 7830. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pp. 221–236.

[3] J. García-Galán, L. Pasquale, P. Trinidad, and A. R. Cortés. "User-centric adaptation of multi-tenant services: preference-based analysis for service reconfiguration". In: *9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2014, Proceedings, Hyderabad, India, June 2-3, 2014*. 2014, pp. 65–74.

[4] E. Gjorven, F. Eliassen, and J. Aagedal. "Quality of Adaptation". In: *Autonomic and Autonomous Systems, 2006. ICAS '06. 2006 International Conference on*. 2006, pp. 9–9.

[5] M. C. Huebscher and J. A. McCann. "A Survey of Autonomic Computing - Degrees, Models, and Applications". In: *ACM Comput. Surv.* 40.3 (2008), 7:1–7:28.

[6] E. Kamsties, F. Kneer, M. Voelter, B. Igel, and B. Kolb. "Feedback-Aware Requirements Documents for Smart Devices". In: *Requirements Engineering: Foundation for Software Quality*. Ed. by C. Salinesi and I. Weerd. Vol. 8396. Lecture Notes in Computer Science. Springer International Publishing, 2014, pp. 119–134.

[7] J. O. Kephart and D. M. Chess. "The Vision of Autonomic Computing". In: *Computer* 36.1 (2003), pp. 41–50.

[8] F. Kneer and E. Kamsties. "Model-based Generation of a Requirements Monitor". In: *Joint Proceedings of REFSQ-2015 Workshops, Research Method Track, and Poster Track co-located with the 21st International Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2015), Essen, Germany, March 23, 2015.* 2015, pp. 156–170.

[9] J. McCann and M. Huebscher. "Evaluation Issues in Autonomic Computing". English. In: *Grid and Cooperative Computing - GCC 2004 Workshops*. Ed. by H. Jin, Y. Pan, N. Xiao, and J. Sun. Vol. 3252. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, pp. 597–608.

[10] "Overview of the Internet of things". In: *IUT-T Y.2060* (2012).

[11] G. G. Pascual, M. Pinto, and L. Fuentes. "Run-time adaptation of mobile applications using genetic algorithms". In: *Proceedings of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2013, San Francisco, CA, USA, May 20-21, 2013*. 2013, pp. 73–82.

[12] A. Ramirez, A. Jensen, and B. Cheng. "A taxonomy of uncertainty for dynamically adaptive systems". In: *Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2012 ICSE Workshop on*. 2012, pp. 99–108.

[13] V. E. Silva Souza, A. Lapouchnian, W. N. Robinson, and J. Mylopoulos. "Awareness Requirements for Adaptive Systems". In: *Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. SEAMS '11. Waikiki, Honolulu, HI, USA: ACM, 2011, pp. 60–69.