Optimization Approaches for Self-Adaptive Systems

Tim Engbrocks

Institute for Program Structures and Data Organization (IPD)
Advisor: Dipl.-Inform. Martina Rapp

- Introduction ist gut

1 Introduction

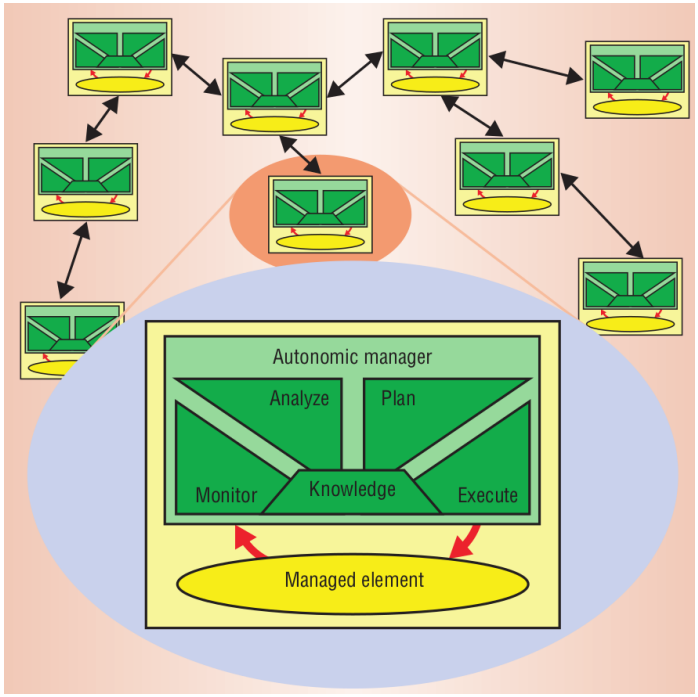

Figure 1: The MAPE-K (Monitor-Analyze-Plan-Execute with Knowledge) feedback loop by Kephart and Chess, 2003 [3]

The complexity of modern software systems is constantly growing. Most of this growth in complexity stems from the "need to integrate several heterogeneous environments into corporate-wide computing systems, and to extend that beyond company boundaries into the Internet" (Kephart and Chess, 2003 [3]). This has reached a state where the "complexity appears to be approaching the limits of human capability" (Kephart and Chess, 2003 [3]). In combination with the uncertainty about a software systems future operations and environment, that the developers of such complex systems face, it becomes uneconomical to purely operate such a system by human operators.

From this the need for software systems which can autonomously manage themselves arises. In order to achieve this task of autonomous self-management, the system has to be able to:

- detect faults and changes in its environment.
- decide how to react to faults and changes in the systems environment.
- make changes to itself.

To model these abilities Kephart and Chess developed the MAPE-K (Monitor-Analyze-Plan-Execute with Knowledge) feedback loop [3] in Figure 1.

Monitor First the system has to monitor itself and its environment.

Analyze The data, gathered by the monitoring step, has to be analyzed to detect changes and faults.

Plan If the analyzing step detects, that an adaptation is necessary, the planning step plans which changes have to be made.

Execute After the changes have been planned, they need to be executed.

Knowledge All of this happens with Knowledge of the environment and the system.

Software systems that can autonomously manage themselves are called Self-Adaptive Systems because of their ability to adapt themselves.

While Self-Adaptive Systems are better at handling more complex systems, human operators are better at handling uncertainty. Because the rules and policies, used by Self-Adaptive Systems, are statically created at the design time of the system, a Self-Adaptive System can adapt its underlying system to a changing environment, but it can not adapt its own adaptation process. This leads to an increasing divergence between the expected results of adaptations and the actual results, when the environment changes in ways that were not predicted by developers during the design time of the system.

From this the need for optimizations for Self-Adaptive Systems arises. While there already are many Optimization Approaches for Self-Adaptive Systems, there is no classification for these Optimization Approaches. Because of this, the existing Optimization Approaches can not be easily compared and it is difficult to identify areas that require further research. This paper aims to provide such a classification for Optimization Approaches for Self-Adaptive Systems

To derive a classification for Optimization Approaches for Self-Adaptive Systems, the second chapter will first explain how Self-Adaptive Systems are classified using three different approaches. Based on these approaches, a classification for Optimization Approaches for Self-Adaptive Systems will be derived and explained in the third chapter. The fourth chapter applies the classification to some existing Optimization Approaches and compares them using the classification.

2 Classification of Self-Adaptive Systems

There are different approaches on how to classify and describe Self-Adaptive Systems, which all focus on different usages, the three approaches that will be highlighted by this paper are:

- FORMS from Weyns et al., 2012 [7]
- Berns and Ghosh, 2009 definition of self-* properties [1]
- Krupitzer's et al., 2015 taxonomy for Self-Adaptive Systems [4]

All of these approaches contain ideas that will be used to construct the classification for Optimization Approaches for Self-Adaptive Systems in the next chapter.

In their 2012 paper FORMS [7] Weyns et al. propose a formal reference model for describing Self-Adaptive Systems. The goal of FORMS is to provide a well defined basis for talking and reasoning about Self-Adaptive Systems. This is achieved by providing a definition of FORMS, which classifies Self-Adaptive Systems, using Z notation. By composing a Self-Adaptive System from different levels of components called subsystems, where each level can adapt the level beneath, FORMS models the self-adaptive process using reflection. The bottom level is populated by base components and domain models. On top of these base components are reflective components and reflection models. The relation between these component or primitives is depicted in Figure 2.

The concept of different layers of reflection will be useful later to distinguish Optimization Approaches for Self-Adaptive Systems from Self-Adaptive Systems.

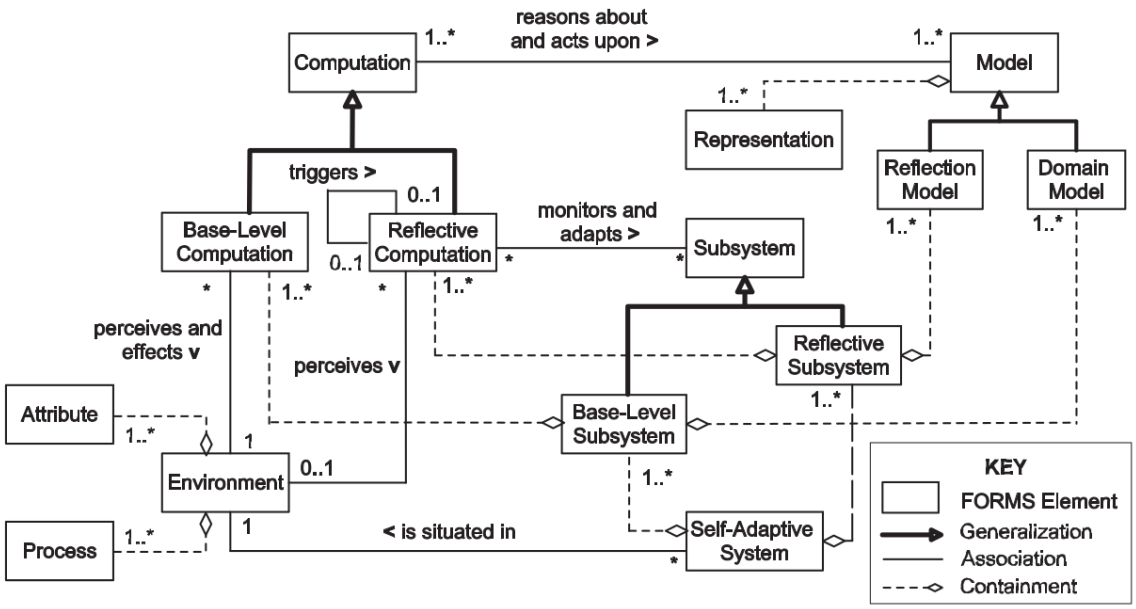

Figure 2: FORMS primitives by Weyns et al., 2012 [7]

SUMMARY OF SELF-* PROPERTIES. F = SET OF EXTERNAL ACTIONS, P = PREDICATE OVER GLOBAL STATE.

Self-* Property Name	**Adversarial actions**	**Behavior**	**Predicate maintained**
Self-stabilization	All transient failures	Restore	P
Self-adapting	Change of environment R	Restore	**if** R_i **then** P_i
Self-healing	Some $C \subseteq F$	Restore	P
Self-organizing	Process join or leave	Maintain, improve, or restore	P
Self-protecting	Some mailicious actions	Maintain	Predicate over trust
Self-optimization	Some $C \subseteq F$	Improve or maximize/minimize as appropriate	An objective function of the global or local state
Self-configuration	Some $C \subset F$, often includes user demands for service	Maintain, improve, or restore using configuration changes	Predicate over system configuration to optimize performance. Sometimes addresses geometric invariant
Self-scaling	Changes of system scale or demand	Restore, or improve	Predicate over service quality
Self-immunity	Some $C \subseteq F$	Eventually maintain	P
Self-containment	Some mailicious actions	Maintain (for a subset of processes)	Predicate over trust

Figure 3: self-* properties by Berns and Ghosh, 2009 [1]

While some of the first ideas for Self-Adaptive Systems focused on self-adaptation, like "The Vision of Autonomic Computing" by Kephart and Chess [3], there are other aspects of software systems that can benefit from the ideas of Self-Adaptive Systems. These aspects are called self-* properties. Berns and Ghosh, 2009 [1] identified and described the self-properties in Figure 3. These self-* properties can be useful to either quickly communicate the abilities and goals of a Self-Adaptive System or to establish well-defined goals for the system.

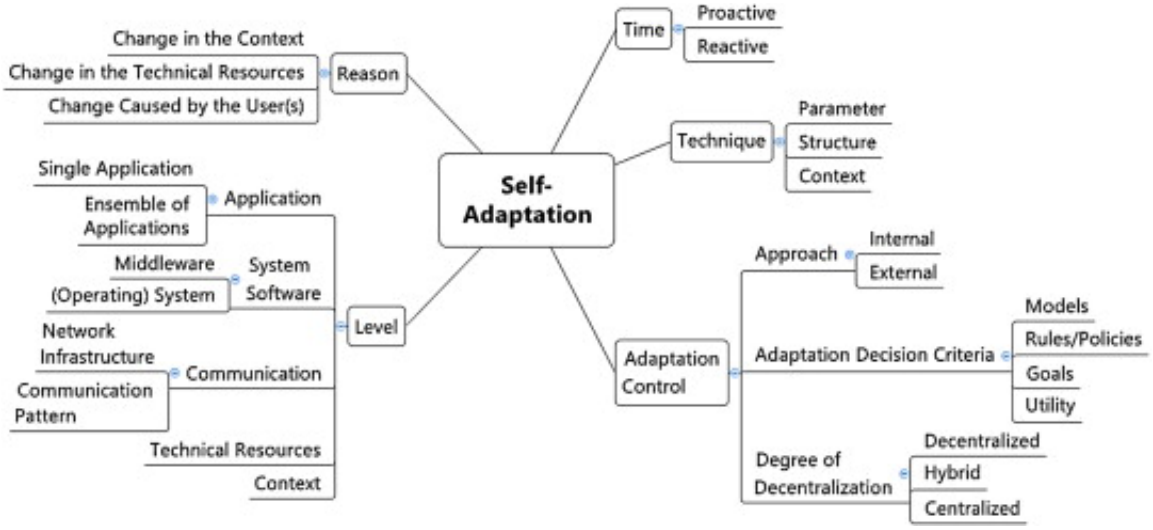

Figure 4: Taxonomy for Self-Adaptive Systems by Krupitzer et al., 2015 [4]

The taxonomy for Self-Adaptive Systems by Krupitzer's et al., 2015 [4] in Figure 4 is based upon the 5W+1H questions by Salehie and Tahvildari, 2009 [5]. These questions are: Where, When, What, Why, Who and How. Each of these questions is responsible for a different aspect of Self-Adaptive Systems and corresponds to a dimension of the taxonomy.

Why First there needs to be a reason for a Self-Adaptive System to adapt. Why an adaptation should be performed is answered by the Reason dimension. According to the taxonomy reasons for an adaptation can be changes in either the context, a technical resource or changes caused by the user.

Where The question of where asks on which level of the system changes need to occur. The different levels on which changes can occur include:

- Different levels of applications from the operating system to a user application.
- How systems communicate with each other.
- The technical resources that are needed by the system.
- The context in which the system operates.

When While the original When-Question by Salehie and Tahvildari tries to understand all temporal aspects of Self-Adaptive Systems, including how frequently changes should occur and if they happen continously, the taxonomy only answers the question when changes should be performed. For this purpose the Time dimension differentiates between systems that perform changes proactively or reactively.

What In addition to the question of where and when changes should occur, it is also important to know what changes should occur. There are different techniques, that can be used. The Technique dimension of the taxonomy differentiates between systems that change parameters, their structure or their context.

Who After answering where, when, what and why changes should be performed, it is necessary to select who is responsible for these changes. According to Salehie and Tahvildari it is also important to establish if the changes can be performed fully autonomous or if the involvement of human operators is necessary. The taxonomy does not directly address all of these concerns but states that: "N/A (nature of a SAS leads to an automatic type of adaptation)" (Krupitzer et al., 2015 [4]).

How Lastly, after determining the where, when, what, why and by who, there needs to be a way to perform the required changes. This is answered by asking how the changes should be performed and corresponds to the Adaptation Control. The three main factors of the Adaptation Control are the degree of decentralization, the adaptation decision criteria and the approach taken by the system, which divides Self-Adaptive Systems into those where the adaptation logic is part of the application logic and those with separated adaptation and application logic.

After classifying Self-Adaptive Systems the following question can be asked: Which parts of a Self-Adaptive System can be optimized? To answer this we will start by looking at which parts can not be optimized or do not benefit from optimization.

The first part that can not be optimized is the environment which provides the Reason dimension. While the environment for a Self-Adaptive System can be chosen in a way which is most beneficial for the system and can be influenced by actors, the behavior of the systems environment can generally not be controlled completely.

Another dimension of Self-Adaptive Systems that can not be optimized, or is not useful to optimize, is the Time dimension. This dimension is mostly a design decision on how the system should behave and be constructed. It is also a question of how to handle uncertainty and the level of accepted risk. A proactive system can prevent faults and degradation in Quality-of-Service metrics but it can also predict the wrong changes, which can lead to a situation where the system itself generates faults by reacting in a way that is contradictive to its goals. A reactive system can not prevent faults like a proactive system but its behavior can be much more stable because it only has to react to a change and not also predict that change.

Lastly the question of who is responsible can not be optimized because the taxonomy simply answers it, by referring to the name "Self-Adaptive System" which implies that the system itself is responsible for managing adaptations.

The three remaining dimensions can be optimized or can benefit from being adapted dynamically. These are the Adaptation Control, the Level and the Technique.

The Approach and the Degree of Decentralization used by the Adaptation Control can not be optimized because they are design decisions of how the system is built. But the Adaptation Decision Criteria can be optimized. An optimization of the Adaptation Decision Criteria could for example be to dynamically adapt the rules and policies at runtime to better reflect a changing environment.

Another dimension that can be optimized is the Technique. This can be optimized by changing what gets adapted by the system.

The last dimension that can be optimized is the level, which can be optimized by dynamically changing at which level of the system adaptations should be performed.

3 Proposal for classification of optimization approaches

Using the idea from Weyns et al. 2012 paper FORMS [7] to compose Self-Adaptive Systems from layers of base and reflective components, an Optimization Approach for Self-Adaptive Systems can be thought of as a reflective component in a layer above the Self-Adaptive System. This allows the application of multiple Optimization Approaches and even the optimization of Optimization Approaches. It also establishes a connection between Optimization Approaches and Self-Adaptive Systems and leads to the realization that a classification for Optimization Approaches of Self-Adaptive Systems should be based upon the same principles as the classifications for Self-Adaptive Systems. Another benefit is a clear distinction between Self-Adaptive Systems and Optimization Approaches for Self-Adaptive Systems: While Self-Adaptive Systems and Optimization Approaches are both composed of reflective components and not base components, the Self-Adaptive System adapts base components, while the Optimization Approach adapts other reflective components.

Following Krupitzer's et al, 2015 [4] taxonomy for Self-Adaptive Systems, a classification for Optimization Approaches of Self-Adaptive Systems should answer the 5W+1H questions by Salehie and Tahvildari, 2009 [5]. These questions are: Where, When, What, Why, Who and How?

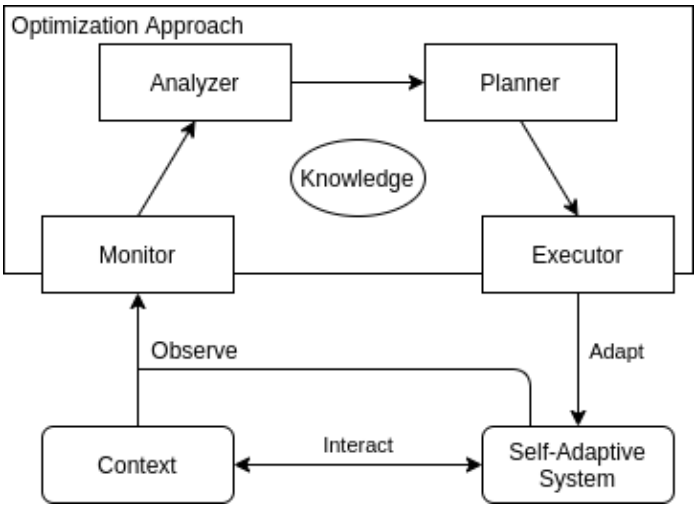

Figure 5: Mapping the optimization process to the MAPE-K feedback loop.

Just like the adaptation process of Self-Adaptive Systems can be understood using the MAPE-K (Monitor-Analyze-Plan-Execute with Knowledge) feedback loop by Kephart and Chess, 2003 [3]. The process of optimizing a Self-Adaptive System can also be expressed using the MAPE-K feedback loop:

- Firstly the optimization approach has to constantly monitor the Self-Adaptive System and its context.
- The data gathered from monitoring can then be analyzed to decided wether or not an optimization is necessary and what should be optimized.

- After deciding that something should be optimized, there needs to be a plan on how to optimize.
- Lastly the planned optimization can be executed.
- For all of this the optimization approach requires knowledge about the Self-Adaptive System and its context.

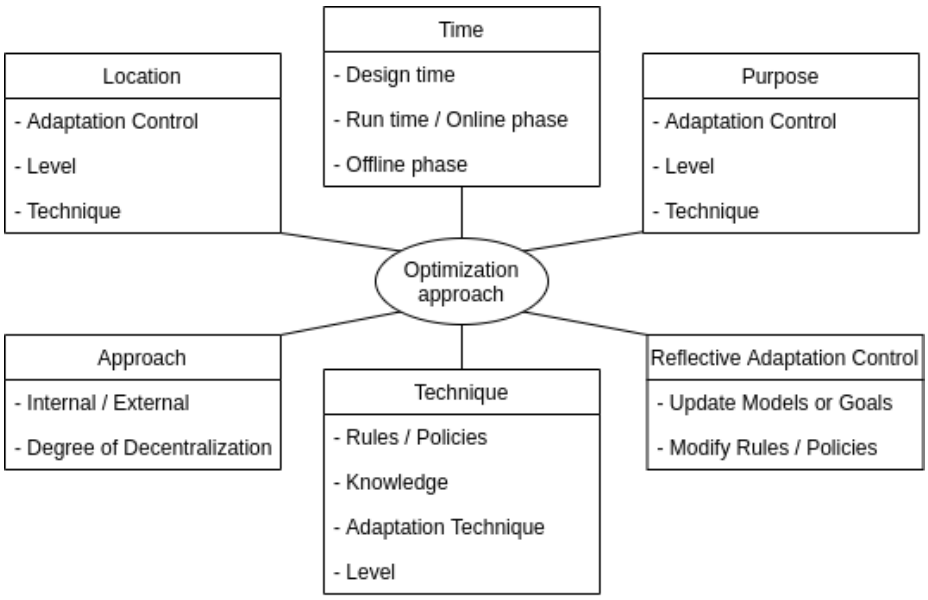

Figure 6: The proposed classification for Optimization Approaches for Self-Adaptive Systems

Location Where in the system is an optimization necessary?
There are three main parts in a Self-Adaptive System that can require optimizations during its lifetime. These are:

- The Adaptation Control.
- The Level on which adaptations are performed
- The Technique that is used to perform adaptations.

Time When are optimizations performed?
Similarly to Self-Adaptive Systems, Optimization Approaches can be differentiated by comparing when they perform optimizations. There are three different phases during the lifetime of a Self-Adaptive System where optimizations can occur. These are:

- At the runtime of the system, which can also be called the online phase.
- During the design time of the system.
- While training the system or during its offline phase.

The training phase can happen in parallel to the run and design time of the system. Examples for this would be when the initial parameters for a Self-Adaptive System are chosen by training a domain model or when generating a new model for the system by training an updated domain model parallel to the running system.

Technique What gets changed to perform the optimization?
What gets changed in a Self-Adaptive System relates to where an optimization is necessary. Because of this, the following parts can be changed:

- The Rules / Policies used by the Adaptation Decision Criteria.
- The Knowledge of the system, which for example can consist of domain models.
- The Adaptation Technique used by the Self-Adaptive System.
- The Level on which the Self-Adaptive System performs changes.

One could argue that the Goals or Utility functions of a Self-Adaptive System could also be changed to optimize its performance. In most cases this is not advisable because Goals and Utility functions can encode information like legal or safety parameters, of which the system has no intrinsic knowledge.

Purpose Why should an optimization occur?
This is perhaps the most important question for any Optimization Approach because it determines if an optimization is necessary. There can be several reasons for performing an optimization in a Self-Adaptive System:

- The system learned new information about for example its environment and needs to update its domain models.
- Because of external or internal changes, a Goal might not be satisfiable anymore.
- The difference of the actual versus the expected outcome of a rule / policy has exceeded a threshold.
- A Utility function needs to be maximized or minimized.

Approach Who is responsible for performing the optimization?

- Is there an internal component that performs changes or are they performed by an external actor.
- Which degree of decentralization is used? Is each component responsible for itself (fully decentralized), are all components optimized by a central entity (fully centralized) or is a hybrid approach used?

Reflective Adaptation Control How is the optimization applied to the system?
Just like the Self-Adaptive System can be described by its Adaptation Control, which is responsible for applying changes to the system, an optimization approach can also be described by how it applies its optimizations to the system. In reference to FORMS by Weyns et al., 2012[7], which uses reflective operations to change components in the system,

the Adaptation Control of the optimization approach will be called Reflective Adaptation Control.

Unlike the Adaptation Control of Self-Adaptive Systems, the Reflective Adaptation Control does not have a dimension for the Approach, because it has its own separate dimension.

- Adaptation Decision Criteria:
  - Models: The system updates its models after receiving new information about its models.
  - Rules / Policies: If the system detects, that it is in a state for which an adaptation rule/policy exists, it should perform that adaptation.
  - Goals: Adaptation should be performed if the system does not meet pre-defined goals.
  - Utility: An adaptation occurs to maximize or minimize a utility function.

Because the proposed classification is based upon the MAPE-K feedback loop and the 5W+1H questions, it is useful to map the classification to each of these models.

MAPE-K	Optimization Approach
Monitor	
Analyze	Location
Plan	Technique
Execute	Reflective Adaptation Control

Figure 7: How the MAPE-K feedback loop by Kephart and Chess, 2003[3] relates to the dimensions of the classification for Optimization Approaches for Self-Adaptive Systems.

5W+1H questions	Optimization Approach
Where	Location
When	Time
What	Technique
Why	Purpose
Who	Approach
How	Reflective Adaptation Control

Figure 8: How the 5W+1H questions by Salehie and Tahvildari, 2009 [5] relate to the dimensions of the classification for Optimization Approaches for Self-Adaptive Systems.

4 Classifying existing optimization approaches

FUSION by Elkhodary et al, 2010 [2] uses a learning and an adaptation cycle. The adaptation cycle corresponds to the Self-Adaptive part of the system and does not directly affect the learning cycle. The learning cycle is the optimization approach used by FUSION, it changes the behavior of the adaptation cycle. Both of these cycles continuously run in parallel to each other.

- Location: Adaptation Control + Technique ?
- Time: FUSION uses both the design and run time to perform optimizations. The design time is used to generate an initial model for the learning cycle. During the run time the learning cycle monitors adaptations and updates the Knowledge Base to improve future adaptations.
- Technique:
- Purpose: The goal of FUSION is to
- Approach: FUSION acts as an external actor.
- Reflective Adaptation Control:

Learning Revised Models for Planning in Adaptive Systems by Sykes et al., 2013 [6]

- Location:
- Time: run time
- Technique:
- Purpose: improve domain model + generate new rules
- Approach: internal component that sits between the domain model and the SAS
- Reflective Adaptation Control:

5 Conclusion

References

[1] Andrew Berns and Sukumar Ghosh. "Dissecting Self-* Properties". In: *2009 Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems*. 2009, pp. 10–19. DOI: 10.1109/SASO.2009.25.

[2] Ahmed Elkhodary, Naeem Esfahani, and Sam Malek. "FUSION: A Framework for Engineering Self-Tuning Self-Adaptive Software Systems". In: *Proceedings of the Eighteenth ACM SIGSOFT International Symposium on Foundations of Software Engineering*. FSE '10. Santa Fe, New Mexico, USA: Association for Computing Machinery, 2010, pp. 7–16. ISBN: 9781605587912. DOI: 10.1145/1882291.1882296. URL: https://doi.org/10.1145/1882291.1882296.

[3] J.O. Kephart and D.M. Chess. "The vision of autonomic computing". In: *Computer* 36.1 (2003), pp. 41–50. DOI: 10.1109/MC.2003.1160055.

[4] Christian Krupitzer et al. "A survey on engineering approaches for self-adaptive systems". In: *Pervasive and Mobile Computing* 17 (2015). 10 years of Pervasive Computing' In Honor of Chatschik Bisdikian, pp. 184–206. ISSN: 1574-1192. DOI: https://doi.org/10.1016/j.pmcj.2014.09.009. URL: https://www.sciencedirect.com/science/article/pii/S157411921400162X.

[5] Mazeiar Salehie and Ladan Tahvildari. "Self-Adaptive Software: Landscape and Research Challenges". In: *ACM Trans. Auton. Adapt. Syst.* 4.2 (May 2009). ISSN: 1556-4665. DOI: 10.1145/1516533.1516538. URL: https://doi.org/10.1145/1516533.1516538.

[6] Daniel Sykes et al. "Learning revised models for planning in adaptive systems". In: *2013 35th International Conference on Software Engineering (ICSE)*. 2013, pp. 63–71. DOI: 10.1109/ICSE.2013.6606552.

[7] Danny Weyns, Sam Malek, and Jesper Andersson. "FORMS: Unifying Reference Model for Formal Specification of Distributed Self-Adaptive Systems". In: *ACM Trans. Auton. Adapt. Syst.* 7.1 (May 2012). ISSN: 1556-4665. DOI: 10.1145/2168260.2168268. URL: https://doi.org/10.1145/2168260.2168268.