Optimization Approaches for Self-Adaptive Systems

Tim Engbrocks

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

1 Introduction

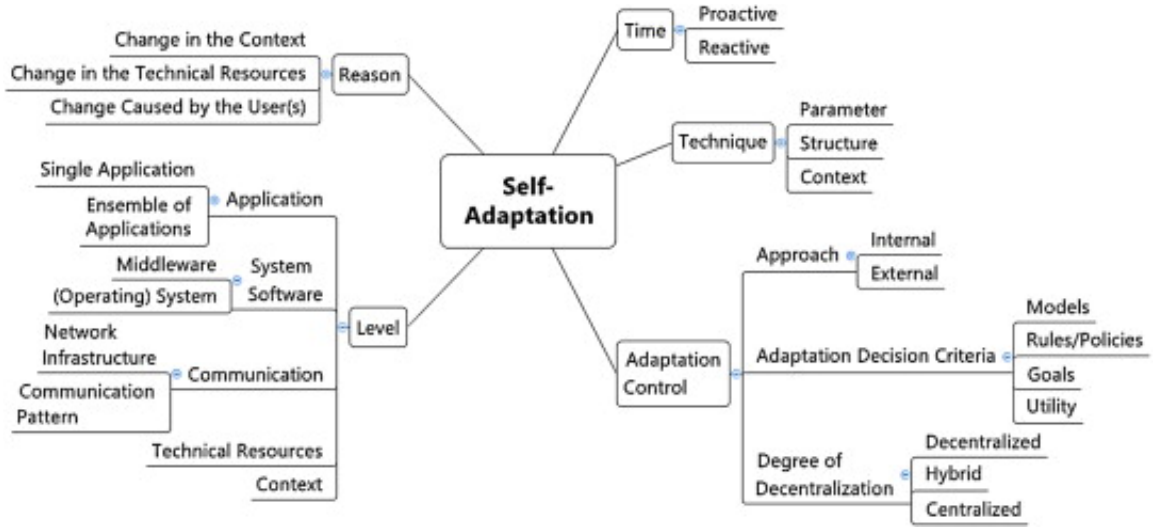

Figure 1: Taxonomy for Self-Adaptive Software by Krupitzer et al., 2015[2]

2 Classification of Self-Adaptive Systems

There are different approaches on how to classify and describe Self-Adaptive Software, which all focus on different usages, like:

- FORMS from Weyns et al., 2012[5] which is a formal reference model for describing Self-Adaptive Software using Z notation.
- Krupitzer's et al., 2015[2] taxonomy for Self-Adaptive Software
- Raibulet's, 2018[3] taxonomy for self-* properties

In this paper Krupitzer's et al., 2015[2] taxonomy for Self-Adaptive Software will be used to classify SAS.

The taxonomy in Figure1 is based upon the 5W+1H questions by Salehie and Tahvildari, 2009[4]. These questions are: Where, When, What, Why, Who and How. The question of who is responsible for an adaptation is trivially answered by the software itself. For each of the remaining questions Krupitzer's et al., 2015[2] taxonomy offers a dimension:

3 Proposal for classification of optimization approaches

When thinking about how to classify optimization approaches for Self-Adaptive Systems, one must realize that an optimization approach for Self-Adaptive Systems is really just a Self-Adaptive System of a Self-Adaptive System.

Because of this the proposed classification for optimization approaches for Self-Adaptive Systems should be based upon the same principles as the classification for Self-Adaptive Systems and it should use the same terminology.

Following Krupitzer's et al, 2015[2] taxonomy for Self-Adaptive Software, a classification for Optimization Approach of SAS should answer the 5W+1H questions by Salehie and Tahvildari, 2009[4]. These questions are:

- Where is the need for change?
- When should a change occur?
- What should be changed?
- Why should something be changed?
- Who should change something?
- How should something be changed?

The process of optimizing Self-Adaptive Software can be split into multiple parts using the MAPE-K (Monitor-Analyze-Plan-Execute with Knowledge) feedback loop by Kephart and Chess, 2003[1]. Each part of the MAPE-K feedback loop can be mapped to the process of optimization in the following way:

- 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.

Location Where in the system is an optimization necessary?

- Adaptation control
- Level
- Technique
- Time

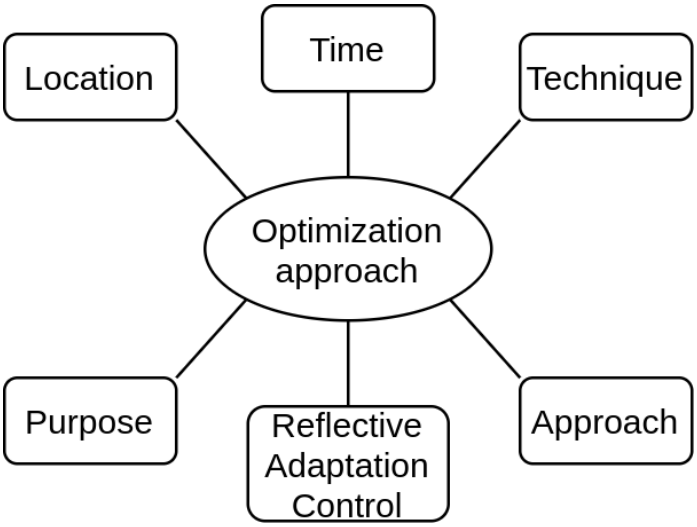

Figure 2: The proposed classification for Optimization Approaches for Self-Adaptive Systems

Time When are optimizations performed?
Similarly to SAS, Optimization Approaches can be differentiated by comparing when they perfom 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
- during the design time of the system
- while training the system

The training phase can happen in parallel to the run and design time of the system. Examples for this would be when the inital parameters for a Self-Adaptive System are choosen 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?

- —

Purpose Why should an optimization occur?
The most important question for any optimization is: What influences the need for an optimization. Just like with Self-Adaptive Systems this can be caused by changes in context, the system itself or updated requirements for the system. Additionally an optimization might be necessary if for example the system detects that its current adaptation control can be improved.

- Changes in context, the system itself or requirements for the system.
- Detection of suboptimal adaptation logic.

Approach Who is responsible for performing the optimization?

- An internal component of the system
- An external actor

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[5], which uses reflective operations to change components in the system, the Adaptation Control of the optimization approach will be called Reflective Adaptation Control.

- —

MAPE Mapping

MAPE	OA
Monitor	
Analyze	
Plan	Technique
Execute	Reflective Adaptation Control

5W+1H Mapping

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

4 Classifying existing optimization approaches

Optimization approach	Purpose	Time	Optimizer		
			Technique	Location	Concern

5 Conclusion

References

[1] 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.

[2] 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.

[3] Claudia Raibulet. "Towards a Taxonomy for the Evaluation of Self-* Software". In: *2018 IEEE 3rd International Workshops on Foundations and Applications of Self* Systems (FAS*W)*. 2018, pp. 22–23. DOI: 10.1109/FAS-W.2018.00020.

[4] 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.

[5] 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.