If a System is Learning to Self-adapt, Who's Teaching?

Yehia Elkhatib
School of Computing Science
University of Glasgow, UK
{*first.last*}@glasgow.ac.uk, ORCID 0000-0003-4639-436X

Abdessalam Elhabbash
School of Computing & Communications
Lancaster University, UK
{*i.last*}@lancaster.ac.uk, ORCID 0000-0002-9463-0446

***Abstract*—Self-adaptation is increasingly driven by machine-learning methods. We argue that the ultimate challenge for self-adaptation currently is to retain the human in the loop just enough to ensure sound evolution of automated self-adaptation.**

***Index Terms*—self-adaptation, self-adaptive systems, autonomous systems, machine learning, deep learning**

I. Introduction

A Self-Adaptive System (SAS) is one that can work around changes on its own. A change could either be a modification in the context that the application runs in, or one in the fabric of the application itself. Some changes might be anticipated by the application designer, such as variations in input data or network connectivity. Other changes are unanticipated, such as unexpected user input or a complete failure of supporting microservices. Handling such unanticipated changes is a big challenge, but *not* the ultimate one for the self-adaptation community. The focus on accelerating system autonomy through the use of machine learning (ML) approaches may undermine the consideration of qualitative aspects (*e.g.,* ethics, safety, and values) in the adaptation loop. Those qualitative aspects need human involvement for satisfying adaptation decisions.

Key Point 1. *We contend that the ultimate challenge for self-adaptation is in controlling automated self-adaptation by involving human users.*

II. Learning to adapt

There are different approaches to self-adaptation. Traditionally, the application designer defines the *adaptation space*, *i.e.,* the range of possible options of adaptation. They also analyze it prior to deployment so that adaptation during runtime would take place using deterministic means such as finite-state machines (*e.g.,* [1], [2]). Recently, automation through ML methods has been introduced as an alternative to either analyze the adaptation space (*e.g.,* [3], [4]), identify the adaptation decision to take (*e.g.,* [5]), or both (*e.g.,* [6]).

In Table I, we present a coarse categorization of deterministic and ML-based adaptation approaches. Research efforts to date have mostly made use of parametric adaptation (*i.e.,* the first row in the table); *cf.* [7]. Based on recent efforts in the literature, we expect: (i) future efforts to cover the other categories, where there is plenty of room for innovation; and (ii) the last category to eventually become the most popular.

Table I
Categorizes of adaptation based on when adaptation space (**AS**) and decision (**D**) are known by the software, where **D** ⊆ **AS**.

Design time	Run time	Adaptation approach
AS & **D**	–	Rule-based automata, Supervised learning
AS	**D**	Semi-supervised / Unsupervised learning
–	**AS** & **D**	Reinforcement learning, Neural networks

Naturally, automated adaptation is pushing to take the human out of the loop, and is indeed successful in doing so; cf. Rogério de Lemos talk at SEAMS 2020 [8]. However, it has introduced a new interaction model between the knowledge held by the human and that acquired by the machine. This issue has been recognized in other fields (*e.g.,* computer vision [9], web search [10]) as the *automation paradox*, which states that with more automation human intervention becomes increasingly critical.

Our concern in this paper is beyond the preoccupation of *how* to adapt. Instead, we are interested in exploring what would happen as we accelerate towards automation and significantly reduced human involvement as a consequence.

Key Point 2. *We have been focusing too much on adaptation whilst eliminating the post-deployment human input. We argue that the human factor needs to be retained in some form for gaining comprehension and critical steering.*

III. Knowledge interaction surfaces

In light of this, we now explore and rationalize the points at which human intervention needs to be retained. We identify these within 2 main surfaces where knowledge is exchanged between human and machine.

A. Human-to-machine surface

The first interaction surface is the one that transfers knowledge from the human elements of a system into its software fabric. We have had numerous works on how to translate human knowledge into rule- / heuristic-based adaptation mechanisms, but we are only in the dawn of how to capture human knowledge into ML models. This is manifested in the input from the following human stakeholders:

- **Developers**: There are a plethora of software libraries to operate ML models. However, there are not enough means to formalize the interface between such libraries and

software tools that developers commonly use. For example, `scikit-learn` and `TensorFlow` are widely used at the information level to provide enhanced operation through parametric changes or deal with a large scale of input data. However, they are seldom used to change the inner structure of an application. For this, better tools are needed to integrate ML libraries with software design methods.

- **Researchers**: Very few practices around developing ML-based adaptation are shared by those working in this area of research [7], [11], [12] especially in terms of assessing data quality and reproducibly inducing model parameters.
- **Domain experts**: During this time of great growth in ML-based adaptation, it is vital to not exclude those with the most knowledge in any given application domain. Software engineering processes are notoriously internally dominated [13], hence the trend towards user- [14] and stakeholder-centered [15] design practices. There is strong evidence of how effective this could be, but efforts have so far been around end-user interaction not system architecture.

Key Point 3. *The interface between human knowledge (in collective and specialized forms) and automated system adaptation is not sufficiently fluid. There is room to innovate by developing software tools and rigorous community practices.*

B. Machine-to-human surface

Once adaptation is automated, human intervention is still needed for different reasons [16]. Model coefficients could of course be tuned in an automated fashion, but other parameters are considered outside the remit of the synthetic knowledge of a SAS. These include, but are not limited to, the following:

- **Early detection**: ML methods in general still suffer from fundamental problems such as overfitting and lack of transparency. This has been observed generally, but also in ML-based adaptation research [7], [12], [17]. Despite the various mitigation proposals (*e.g.,* regularization, cross-validation, etc.), humans are still quicker and more effective in identifying and resolving such issues [18], [19].
- **Data quality**: For example "the road markings are too patchy" for a self-driving vehicle in, say, a rural part of a low-income country, or "the workforce is on strike" for a factory assembly line controller.
- **The unmeasurables**: In designing ML-based systems, we tend to gravitate towards what is readily measurable (primarily performance and reliability). However, some aspects are concretely human and cannot (yet) be reasoned about using software alone; *e.g.,* ethics, equality, and inclusivity.
- **Significant decision**: Some decisions still need to be made by a human; such as "the patient is pronounced dead" or "the enemy has attacked". Although a SAS can provide significantly enriching information to feed such decisions, a human is still better at making such decisions that the current generation of ML-based SASs.
- **Meta reflection**: Human perspective is important for regularly checking if the adaptation goals are still valid at a high level and if the models reflect them accurately. We are not expecting systems to always be explainable as this is a long-term goal, but they should at least be presentable in modifiable forms to the developer, maintainer, and end-user.

Key Point 4. *Human intervention is still needed to guide the evolution of adaptation at certain key points.*

IV. Conclusion

In summary, we believe that the ultimate challenge for self-adaptation is not in handling unanticipated changes but in controlling automated self-adaptation by involving human users. The knowledge exchange processes we discuss here highlight *why* and *what* interfaces are needed, and *who* for. In turn, the interfaces will answer the *how*, presenting a whole host of great research opportunities. Several other questions remain: *when* and *where* does it make sense to solicit human input; *how much* depending on expertise, mental state, etc.

References

[1] P. Arcaini, E. Riccobene, and P. Scandurra, "Modeling and analyzing MAPE-K feedback loops for self-adaptation," in *SEAMS*, 2015.

[2] V. Klös, T. Göthel, and S. Glesner, "Comprehensible and dependable self-learning self-adaptive systems," *Journal of Systems Architecture*, vol. 85-86, pp. 28–42, 2018.

[3] J. Van Der Donckt, D. Weyns, F. Quin, J. Van Der Donckt, and S. Michiels, "Applying deep learning to reduce large adaptation spaces of self-adaptive systems with multiple types of goals," in *SEAMS*, 2020.

[4] K. Mori, N. Okubo, Y. Ueda, M. Katahira, and T. Amagasa, "Supporting viewpoints to review the lack of requirements in space systems with machine learning," in *SEAMS*, 2020.

[5] D. Weyns and M. U. Iftikhar, "ActivFORMS: A model-based approach to engineer self-adaptive systems," *arXiv CoRR*, vol. 1908.11179, 2019.

[6] A. Metzger, C. Quinton, Z. Á. Mann, L. Baresi, and K. Pohl, "Feature-model-guided online learning for self-adaptive systems," *arXiv CoRR*, vol. 1907.09158, 2019.

[7] B. Porter, R. R. Filho, and P. Dean, "A survey of methodology in self-adaptive systems research," in *ACSOS*, 2020.

[8] R. de Lemos, "Human in the loop: What is the point of no return?" in *SEAMS*, 2020.

[9] A. Cummaudo, R. Vasa, J. Grundy, M. Abdelrazek, and A. Cain, "Losing confidence in quality: Unspoken evolution of computer vision services," in *ICSME*, 2019.

[10] C.-J. M. Liang *et al.*, "AutoSys: The design and operation of learning-augmented systems," in *USENIX ATC*, 2020.

[11] V. Riccio, G. Jahangirova, A. Stocco, N. Humbatova, M. Weiss, and P. Tonella, "Testing machine learning based systems: a systematic mapping," *Empir. Softw. Eng.*, vol. 25, no. 6, pp. 5193–5254, 2020.

[12] O. Gheibi, D. Weyns, and F. Quin, "Applying machine learning in self-adaptive systems: A systematic literature review," *arXiv CoRR*, vol. 2103.04112, 2021.

[13] D. Renzel, I. Koren, R. Klamma, and M. Jarke, "Preparing research projects for sustainable software engineering in society," in *ICSE Software Engineering in Society Track*, 2017.

[14] A. Seffah and A. Andreevskaia, "Empowering software engineers in human-centered design," in *ICSE*, 2003.

[15] H. Cinis, "Improving the definition of software development projects through design thinking led collaboration workshops," in *ICSE Software Engineering in Practice*, 2018.

[16] M. Gil, V. Pelechano, J. Fons, and M. Albert, "Designing the human in the loop of self-adaptive systems," in *Ubiquitous Computing and Ambient Intelligence*. Springer, 2016, pp. 437–449.

[17] M. Scheerer, J. Klamroth, R. Reussner, and B. Beckert, "Towards classes of architectural dependability assurance for machine-learning-based systems," in *SEAMS*, 2020.

[18] P. Domingos, "A few useful things to know about machine learning," *Commun. ACM*, vol. 55, no. 10, p. 78–87, Oct. 2012.

[19] C. Zhang, O. Vinyals, R. Munos, and S. Bengio, "A study on overfitting in deep reinforcement learning," *arXiv CoRR*, vol. 1804.06893, 2018.