A Concept for Self-Explanation of Macro-Level Behaviour in Lifelike Computing Systems

Martin Goller[1] and Sven Tomforde[1]

[1]Intelligent Systems, Christian-Albrechts-Universität zu Kiel, Germany
goller.cau@gmail.com / st@informatik.uni-kiel.de

Abstract

The basic idea of developing future 'lifelike' systems is to transfer qualities in technical utilisation that go beyond well-established mechanisms such as self-adaptation, learning, and robustness. In this paper, we argue that the resulting systems will need components to self-explain their behaviour - if we want to avoid acceptance issues that result from surprising and irritating system behaviour. Such self-explaining behaviour needs to answer the questions of when, what and how explanations should be provided to the user. We review existing metrics, outline a concept of how to address the 'when' question and identify corresponding research challenges towards an automated generation of explanations.

I. Introduction

Recent trends in information and communication technology entailed increasingly autonomous systems that adapt their own behaviour and try to optimise it over time – resulting in so-called self-adaptive and self-organising (SASO) systems. Initiatives such as Organic (Müller-Schloer and Tomforde (2017)) and Autonomic Computing (Kephart and Chess (2003)) are concrete manifestations and pioneers of this trend, which is supported by new applications such as autonomous driving (Levinson et al. (2011)) or the Internet of Things (Weber and Weber (2010)). SASO technology is understood as an approach to keeping the complexity of increasingly integrated, open and dynamic systems manageable, as it is no longer possible to plan all possibilities in advance at design time. At the same time, the integration of machine learning methods is intended to create novel possibilities to react appropriately to the unknown and at the same time continuously strive for better behaviour.

Even though SASO technology already had its roots in cybernetics and has been drastically strengthened again in the last two decades (perhaps starting from Tennenhouse's Proactive Computing, see (Tennenhouse (2000)), and Weiser's vision of Pervasive Computing, see (Weiser (1999))), we can state that controllability and reacting or adapting to the unknown remain the central challenges. This realisation leads, among other things, to the approach of making technical systems even more lifelike, which e.g. takes up and continues the original motivation of the OC and AC initiatives.

In this article, we note that in addition to the obvious lifelike properties such as evolution and continuous adaptation to the 'living space' or 'environmental niche' as well as focused response mechanisms (to name only the obvious examples), another prominent challenge comes to the fore in the acceptance of such systems by the human users (or better: stakeholders or influenced persons). This raises the question of how such systems can explain their behaviour to humans in an automated way, which in turn leads directly to the two crucial questions: When are explanations of behaviour necessary? And: What needs to be explained.

From a developer's point of view, this primarily means that we need a concept to answers the question of 'when', which then enables us to answer the 'what'. Therefore, this article explains a concept to measure system behaviour, whereupon abnormal behaviour of these measurements will then serve as an answer to the question 'when'.

Building on recent work, we present a measurement framework for system behaviour that forms the basis for such an explanation framework (Section II). In addition, we discuss possible further variables that can be relevant for lifelike behaviour and can therefore be integrated into the framework. On this basis, we discuss a concept to automatically detect events that serve as triggers for self-explanations, which is combined with a principled, possible use to answer the question 'what' (Section III). Since this is intended as a first concept, we highlight the most urgent research challenges to automatically generate the resulting self-explanations. The article concludes with a summary and an outlook on how the defined concepts can be explored and implemented.

II. A Measurement Framework for Macro-Level Behaviour Assessment

The basis of our approach to self-explanatory mechanisms of lifelike technical systems is the possibility of quantifying system behaviour by means of (external) observation. To this end, in this section, we first present our system model, which we currently assume - and which can form the basis for future lifelike systems. Using this system model, we then explain existing and potential approaches for quantifying system properties.

System Model

In this article, we refer to a technical system S as a collection A of autonomous subsystems a_i that are able to adapt their behaviour based on self-awareness of the internal and external conditions. We further assume that such a subsystem is an entity that interacts with other entities, i.e., other systems, including hardware, software, humans, and the physical world with its natural phenomena. These other entities are referred to as the *environment* of the given system. The *system boundary* is the common frontier between the system and its environment.

Each $a_i \in A$ is equipped with sensors and actuators (both, physical or virtual). Internally, each a_i consists of two parts: The productive system part *PS*, which is responsible for the basic purpose of the system, and the control mechanism *CM*, which controls the behaviour of the PS (i.e., performs self-adaptation) and decides about relations to other subsystems. In comparison to other system models, this corresponds to the separation of concerns between *System under Observation and Control* (SuOC) and *Observer/Controller* tandem (Tomforde et al. (2011)) in the terminology of Organic Computing (OC) (Tomforde et al. (2017b)) or *Managed Resource* and *Autonomic Manager* in terms of Autonomic Computing (Kephart and Chess (2003)). Figure 1 illustrates this concept with its input and output relations. The user describes the system purpose by providing a utility or goal function U which determines the behaviour of the subsystem. The User usually takes no further action to influence the decisions of the subsystem. Actual decisions are taken by the productive system and the CM based on the external and internal conditions and messages exchanged with other subsystems. We model each subsystem to act *autonomously*, i.e., there are no control hierarchies in the overall system. Please note that for the context of this article an explicit local configuration of the PS is necessary – which in turn limits the scope of the applicability of the proposed method. Furthermore, each subsystem must provide read-access to the configuration.

At each point in time, the productive system of each a_i is configured using a vector c_i. This vector contains a specific value for each control variable that can be altered to steer the behaviour, independently of the particular realisation of the parameter (e.g., as real value, boolean/flag, integer or categorical variable). Each subsystem has its own configuration space, i.e. an n-dimensional space defining all possible realisations of the configuration vector. The combination of the current configuration vectors of all contained subsystems of the overall system S defines the joint configuration of S. We assume that modifications of the configuration vectors are done by the different CM only, i.e. locally at each subsystem, and are the result of the self-adaptation process of the CM.

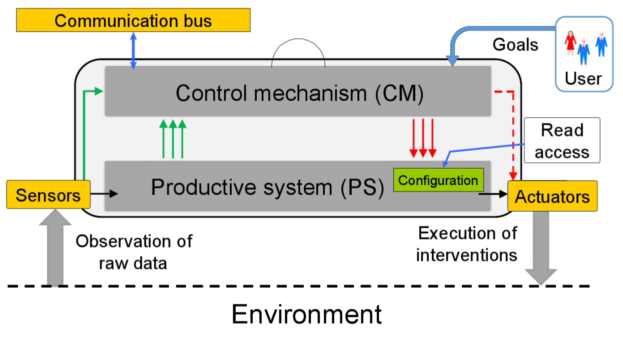

Figure 1: Schematic illustration of a subsystem a_i from (Tomforde et al. (2011)). The arrows from the sensors and PS to the CM indicate observation flows, while the arrows from the CM to the PS and the actuators indicate control flows. Dashed arrows emphasise a possible path that is typically not used. Not shown: The CM is able to communicate with other CMs in the shared environment to exchange information such as sensor reading and to negotiate policies using the communication bus.

This system model describes an approach based on the current state-of-the-art in the field of self-adaptive and self-organising systems. We assume that ongoing research towards more lifelike systems will shift the boundaries in terms of the underlying technology as well as the possibility to alter higher-levelled design decisions – but it will most likely not result in entirely new design concepts. In turn, we assume that fundamental questions will arise about how the CM evolves according to the characteristics of its 'environmental niche', for instance, but the separation of concerns between CM and PS remain visible.

Standard Measures

Traditionally, the success and the behaviour of a technical system is quantified using the primary purpose of the application. In the first place, this directly refers to the system goal. Based on the categorisation proposed by McGeoch in McGeoch (2012), we distinguish between the two performance aspects *quality of the solution* and *time required for the solution*. The latter defines how much time the system required to solve the given purpose or application – where the time can be given in CPU cycles, in real-time, or even in a logical time. Intuitively, such a time-based measure comes with high precision depending on the underlying resolution but it does not include any statements about the quality or generality. In particular, it may depend on the specific hardware equipment that has been used in the experiments. In turn, the quality itself (which may be expressed in a degree of goal achievement) says nothing about the time required to accomplish the goal.

These overarching aspects of system behaviour are augmented with a more theoretical analysis of the applicability. In particular, given techniques such as the O(n) notation, runtime and memory complexity are quantified. This can be extended with verification of processes, i.e., guarantees that may be quantified in terms of coverage or degree of guarantee-able behaviour. As an alternative, the 'restore-invariant approach' by Nafz et al. Nafz et al. (2011) establishes a formal framework for self-organisation behaviour that may serve as a quantifiable basis.

In addition to these considerations, the robustness and resilience of systems, as well as their behaviour, can be quantified using specific metrics. One recent example from the domain of Organic Computing systems can be found in (Tomforde et al. (2018)).

Self-Adaptation-based Measures

Due to the shift in responsibility as visualised by our system model depicted in Figure 1 – a CM is added to the PS that performs (semi-)autonomous decisions – research on SASO systems entailed augmenting the measurement framework by several SASO-specific metrics. For instance, Kaddoum et al. (Kaddoum et al. (2010)) discuss the need to refine classical performance metrics to SASO purposes and present specific metrics for self-adaptive systems. They distinguish between "nominal" and "self-*" situations and their relations: The approach measures the operation time about the adaptation time to determine the effort. This includes aspects such as the adaptation speed to detected changes. Some of the developed metrics have been investigated in detail by Camara et al. for software architecture scenarios (Cámara et al. (2014)). Besides, success and adaptation efforts and ways to measure autonomy have been investigated, see e.g. (Gronau (2016)).

In addition to these goal- and effort-based metrics, several further measurements indicate a macro-level behaviour of a set of autonomous subsystems. The most important are:

a) *Emergence* is basically described as the emergence of macroscopic behaviour from microscopic interactions of self-organised entities (Holland (2000)). In the context of SASO systems, this refers to the formation of patterns in the system-wide behaviour, for instance. Examples for quantification methods are (Mnif and Müller-Schloer (2011)) and (Fernández et al. (2014)).

b) *Self-organisation* can be expressed as a degree to which the autonomous subsystems forming an overall SASO system decide about the system's structure without external control, where the structure is expressed as interaction/cooperation/relation among individual subsystems (Müller-Schloer and Tomforde (2017)). Examples of quantification methods are using a static approach (see (Schmeck et al. (2010)) and methods using a dynamic approach, see (Tomforde et al. (2017a)). An alternative discussion of self-organisation and its relation to emergence is given by De Wolf and Holvoet (2004).

c) *Self-adaptation* refers to the ability of systems to change their behaviour according to environmental conditions, typically with the goal to increase a utility function. The degree of adaptivity can be measured using static (Schmeck et al. (2010)) and dynamic (Tomforde and Goller (2020)) approaches.

d) *Scalability* is a property that defines how far the underlying mechanisms are still promising if the number of participants grows strongly. Quantitatively, this can be expressed as an exponent for the control overhead, for instance.

e) *Stability* is to a certain degree a meta-measure applied, e.g., to the degrees of self-adaptation and self-organisation. It determines how far the metrics are static allowing for standard changes and identifying deviations from the expected behaviour. An example can be found in (Goller and Tomforde (2020)).

f) *Variability* or *Heterogeneity* are terms referring to population of individual subsystems as they focus on the differences in the behaviour, the capabilities or the strategies followed by the subsystems. Examples can be found in (Schmeck et al. (2010)) and (Lewis et al. (2015)).

g) *Mutual influences* among distributed autonomous subsystems indicate that the decisions of one have an impact (e.g., on the degree of utility achievement) of another subsystem (Rudolph et al. (2019)). An example for a quantification technique based on the utilisation of dependency measures is given inRudolph et al. (2016).

Possible Lifelike-oriented Measures

Considering the concept of lifelike technical systems and their desired capabilities, the set of existing metrics is probably not sufficient enough to cover the entire behaviour. In particular, we will have to investigate novel measurement techniques that explicitly cover lifelike attributes. Although there is currently no exact definition of what lifelike computing systems are, we can approach the question of what is missing in the measurement framework by considering 'qualities of life' that we aim to transfer to technical usage and that go beyond the SASO-based scalability, adaptation, organisation, or robustness questions. In particular, we identified the following aspects as primary options based on the considerations of how we consider lifelike systems outlined in Section I:

First, lifelike system will evolve over time which may include an adaptation of its primary usage. Consequently, a first measure should aim at quantifying the evolution behaviour itself and a second one the coverage of the primary purpose. The latter case continues the ideas formulated in the Organic Computing initiative when defining the property of 'flexibility', i.e. how far a SASO system can react appropriately to changing goal functions (Becker et al. (2012)).

Second, this evolution corresponds to an adaptation to the niche in which the system survives. This may be expressed with a measure of 'fitness in the niche' or 'degree of niche appropriateness'.

Third, such an evolution implies that the system is somehow converted (or better: converts itself). Besides the description of this process of time, a more static measure based on the design can aim at determining a 'degree of convertability', i.e. the freedom to which the system can evolve during operation.

Fourth, this may include a transfer to an entirely different niche, or in other words to another problem domain. This can be expressed in a static manner by comparing the current problem space with the initial one or in a dynamic manner by a degree of transfer that the system has undergone.

Fifth, such a lifelike, evolutionary behaviour is done in the context of the environmental conditions, which includes the presence of other subsystems in open system constellations. As a result, parts of the decisions of a lifelike system are about the current integration into such a constellation, resulting in 'self-improving system integration' (Bellman et al. (2021)). Although there is currently no integration measure available, recent work suggests that such an integration state is probably a multi-objective function that builds upon metrics mentioned in the context of SASO measures (Gruhl et al. (2018)).

Finally, such a lifelike character obviously has implications on the way we design and operate systems. In contrast to current practices that take design-related decisions and provide corridors of freedom for the self-* mechanisms, design-time decisions themselves need to become reversible or changeable by the systems, resulting in a degree of reversibility (in a static manner) or changes (in the sense of how strong the design has already been altered).

Obviously, this list is not meant to be complete. It illustrates the need for further techniques that are suitable to quantify the lifelike-based properties. We are convinced that a necessary path in lifelike research is to fill this gap with an integrated measurement framework that provides a basis for comparison and assessment of the observed runtime behaviour.

III. Self-Explanations based on Macro-Level Behaviour Assessment

Within the last year, several contributions proposed steps towards a self-explanation of technical systems, particularly focusing on aspects of self-adaptation. The most prominent examples can be found in Fähndrich et al. (2013) (with a Bayesian reasoning approach), Guidotti et al. (2018) (with a focus on black-box classification), Bencomo et al. (2012) (with a software engineering approach considering the satisfaction of the requirements of a self-adaptive system), Welsh et al. (2014) (also from a software engineering point of view with a focus on accomplishing runtime goals), Klös (2021) (based on an integrated design and verification framework – and the corresponding deviations), or Parra-Ullauri et al. (2020) (based on a multi-level reasoning approach using temporal models).

In contrast to these approaches, we propose to develop a self-explanation component for lifelike technical systems that builds upon the metrics outlined above and establishes an observation and explanation loop. The idea of such a self-explanation is that this should cover the following aspects:

- It should only be provided if the system recognises unanticipated behaviour or abrupt shifts that are perceived by humans that interact with the system (otherwise we face an attention problem of users)
- The explanations should contain information about what changed and why this change happens, which includes the triggers that have been identified as root causes (e.g. a failure of a component, abnormal external effects or behaviour change of other systems)
- This may be augmented with an estimation of the impact and severity as well as a prediction of the upcoming developments.
- Further, the self-explanation should come in a human-understandable format, i.e. using human-interpretable terminology (e.g., 'Device X became too hot due to overload that was caused by new component Y')
- Finally, these explanations have to be generated in a timely and accurate manner and become subject to a learning process that optimises the self-explanation per user. In particular, this can consider direct (i.e., approval or intervention at goal level) and indirect (i.e., recognition and no following action by the user) feedback for optimisation purposes. The result will then be a user-specific degree of explanation behaviour.

Figure 2 illustrates the envisioned process that works as follows:

1. An observation loop is established at the macro-level that gathers all externally visible variables of the contained subsystems. We aim at the maxro- or system-wide level for explanations as we consider the autonomous subsystems as components for the overall functionality. However, this system boundary choice depends on the purpose and the perception of the user.
2. The resulting data is pre-processed, brought into an appropriate representation and analysed to determine the key figures. This includes static and dynamic indicators.
3. Based on novelty/anomaly/change detection such as Gruhl et al. (2021), unexplainable or unanticipated behaviour of these key figures is recognised and assessed. In particular, this should come up with scores for the degree of uncertainty of the observed behaviour (with uncertainty being defined as 'explainable from previously seen

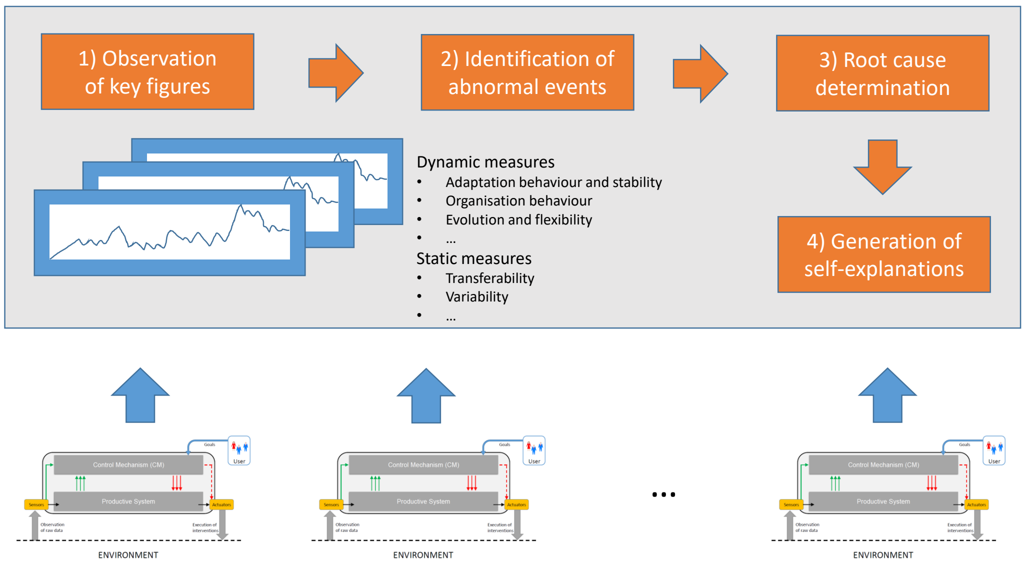
1) Observation of key figures
2) Identification of abnormal events
3) Root cause determination
4) Generation of self-explanations
Dynamic measures
Adaptation behaviour and stability
Organisation behaviour
Evolution and flexibility
...
Static measures
Transferability
Variability
...
Control Mechanism (CM)
Productive System
Sensors
Actuators
Goals
User
Observation of raw data
Execution of interventions
ENVIRONMENT
...

Figure 2: Schematic illustration of the self-explanation process using an external monitoring approach. Based on continuous observation of static (i.e. design properties) and dynamic (i.e. time series) key indicators, unexpected or abnormal behaviour is detected. For such events, possible root causes are identified and ranked according to their plausibility. This serves as input to automatically generate self-explanations to the user that are human-interpretable and indicate i) What happens, ii) what the root causes are and iii) what impact this has on the system behaviour. This may further be enriched with statements about the expected impact or predicted future developments if possible.

behaviour'). Such an abnormal event answers the question 'when should a self-explanation be generated?'

4. As soon as this trigger is found, the states of the contained subsystems and their sequences are analysed to identify possible root causes. Again, this may make use of anomaly detection techniques that consider the different state variables and provide uncertainty values again. These possible root causes are collected, aggregated, correlated, prioritised, resulting in an ordered list of possible causes.

5. Based on this event-to-cause mapping, a self-explanation is generated and provided to the user.

Please note that the integrated quantification framework to assess the system behaviour and the corresponding explanation loop is assumed to work at the macro-level (i.e., without any insight on the specific mechanisms and representations of the individual subsystems), since some of the metrics only occur at macro-level (e.g. emergence). However, this can be turned into a hybrid system approach, where each subsystem cooperates with the system-wide loop to filter, augment, and customise the explanations.

Considering this envisioned process towards automated self-explanations in lifelike systems, we face several research challenges. In the following paragraphs, we outline the most urgent ones and provide first ideas on how to solve them.

Challenge 1 - Metrics: The first challenge is concerned with the metrics briefly summarised in Section II. In particular, we have to answer the questions, which of these metrics is relevant? This includes answers to the question of what do metrics for the quantification of lifelike qualities look like? Based on this, we have to define a mechanism to pre-process, and represent the incoming data – which defines a standard time-series analysis problem.

Challenge 2 - Types of metrics and availability: As outlined above, we distinguish between static and dynamic measures. Considering the inherent heterogeneity, we have to find the concept of how to fuse the measures by integrating both, static and dynamic measures. This further results in questions of how to augment the pure scores, i.e. if predictions of upcoming behaviour are required and in which resolution to allow for proactive actions.

Challenge 3 - Anomaly/Novelty detection: The core of our concept lies in a sophisticated detection of triggers, which is defined as unexpected behaviour of the key indicators. Technically, this should be realised in terms of anomaly, novelty or change detection techniques. Consequently, the questions arises which techniques are most appropriate and how we can provide online methods. Here, we can make use of approaches from the field of self-integrating systems (Gruhl et al. (2021)) that already focus on the desired capabilities.

Challenge 4 - Root cause detection: Given that a trigger for self-explanation is detected, we have to identify the root causes that have been responsible for the observed behaviour. This means to provide techniques that are able to detect possible root causes (also as sequences of interconnected events and not just as isolated events) and rank them? Based on such an approach, we have to investigate how to select the most likely root cause or set/sequence of root causes that explain the behaviour.

Challenge 5 - Definition of explanations: Above, we already mentioned which information an explanation to the user should contain. This needs to be formalised and investigated in detail. In particular, this results in the challenge of how to generate explanations and which aspects they should comprise.

Challenge 6 - Presentation of explanations: Finally, explanations have to be automatically presented to the user in a human-understandable manner. This implies that we have to develop concept of combining human-based terms (such as cold/warm or fast/slow) with system-based measures. Possible concepts could establish joint input spaces and use human feedback to learn the resulting mapping. Using such a joint representation, concepts from deliberative abductive reasoning (Dessalles (2015)) may serve as a basis for these approaches.

IV. Conclusions

In this paper, we outlined our notion of what lifelike technical systems should be - or better which qualities of life we aim at imitating in technical systems that go beyond the well-established field of self-adaptive and self-organising systems. Based on this, we reviewed approaches to quantify system behaviour mainly at the macro-level using a standard system model for self-adaptation. This review also included an identification of possible measurement approaches that close the gap for observation and behaviour assessment of future lifelike systems.

In our notion, an important property of these lifelike systems will be to allow for self-explanation of decisions and resulting behaviour, otherwise the acceptability of even more autonomous and evolving systems will most likely face acceptance problems. We outlined that such self-explanation has to answer two major questions: i) When to provide self-explanations to the user and ii) what is explained (including 'how'). This paper proposed to address the first question by using a measurement framework.

Future work will investigate possible metrics to quantify especially the evolution aspects of lifelike behaviour, including the properties of reversibility or transferability of the system purpose. By using selected applications as use cases, we aim at analysing how the identification of events or conditions that need explanations to the users can be established. Following this, the final goal of this research is to provide mechanisms and techniques that actually derive human-understandable self-explanations.

References

Becker, C., Hähner, J., and Tomforde, S. (2012). Flexibility in organic systems - remarks on mechanisms for adapting system goals at runtime. In *Proc. of 9th Int. Conf. on Inf. in Control, Automation and Robotics*, pages 287–292.

Bellman, K. L., Botev, J., Diaconescu, A., Esterle, L., Gruhl, C., Landauer, C., Lewis, P. R., Nelson, P. R., Pournaras, E., Stein, A., and Tomforde, S. (2021). Self-improving system integration: Mastering continuous change. *Future Gener. Comput. Syst.*, 117:29–46.

Bencomo, N., Welsh, K., Sawyer, P., and Whittle, J. (2012). Self-explanation in adaptive systems. In *2012 IEEE 17th International Conference on Engineering of Complex Computer Systems*, pages 157–166. IEEE.

Cámara, J., Correia, P., de Lemos, R., and Vieira, M. (2014). Empirical resilience evaluation of an architecture-based self-adaptive software system. In *Pro. of 10th Int. ACM Sigsoft Conf. on Quality of Softw. Architectures*, pages 63–72.

De Wolf, T. and Holvoet, T. (2004). Emergence versus self-organisation: Different concepts but promising when combined. In *International workshop on engineering self-organising applications*, pages 1–15. Springer.

Dessalles, J.-L. (2015). A cognitive approach to relevant argument generation. In *Principles and Practice of Multi-Agent Systems*, pages 3–15. Springer.

Fähndrich, J., Ahrndt, S., and Albayrak, S. (2013). Towards self-explaining agents. *Trends in Practical Applications of Agents and Multiagent Systems*, pages 147–154.

Fernández, N., Maldonado, C., and Gershenson, C. (2014). Information measures of complexity, emergence, self-organization, homeostasis, and autopoiesis. In *Guided self-organization: Inception*, pages 19–51. Springer.

Goller, M. and Tomforde, S. (2020). Towards a continuous assessment of stability in (self-)adaptation behaviour. In *2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems, ACSOS 2020*, pages 154–159.

Gronau, N. (2016). Determinants of an appropriate degree of autonomy in a cyber-physical production system. *Proc. of 6th Int. Conf. on Changeable, Agile, Reconfigurable, and Virtual Production*, 52:1 – 5.

Gruhl, C., Sick, B., and Tomforde, S. (2021). Novelty detection in continuously changing environments. *Future Gener. Comput. Syst.*, 114:138–154.

Gruhl, C., Tomforde, S., and Sick, B. (2018). Aspects of measuring and evaluating the integration status of a (sub-)system at runtime. In *2018 IEEE 3rd International Workshops on Foundations and Applications of Self* Systems*, pages 198–203.

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., and Pedreschi, D. (2018). A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):1–42.

Holland, J. H. (2000). *Emergence: From chaos to order*. OUP Oxford.

Kaddoum, E., Raibulet, C., Georgé, J.-P., Picard, G., and Gleizes, M.-P. (2010). Criteria for the evaluation of self-* systems. In *Pro. of ICSE Works. on Softw. Eng. for Adaptive and Self-Managing Sys.*, pages 29–38.

Kephart, J. and Chess, D. (2003). The Vision of Autonomic Computing. *IEEE Computer*, 36(1):41–50.

Klös, V. (2021). Safe, intelligent and explainable self-adaptive systems. *PhD thesis, Technical University Berlin, Germany*.

Levinson, J., Askeland, J., Becker, J., Dolson, J., Held, D., Kammel, S., Kolter, J. Z., Langer, D., Pink, O., Pratt, V., et al. (2011). Towards fully autonomous driving: Systems and algorithms. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 163–168. IEEE.

Lewis, P. R., Esterle, L., Chandra, A., Rinner, B., Torresen, J., and Yao, X. (2015). Static, dynamic, and adaptive heterogeneity in distributed smart camera networks. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 10(2):1–30.

McGeoch, C. C. (2012). *A guide to experimental algorithmics*. Cambridge University Press.

Mnif, M. and Müller-Schloer, C. (2011). Quantitative emergence. In *Organic Computing—A Paradigm Shift for Complex Systems*, pages 39–52. Springer.

Müller-Schloer, C. and Tomforde, S. (2017). *Organic Computing – Technical Systems for Survival in the Real World*. Autonomic Systems. Birkhäuser Verlag.

Nafz, F., Seebach, H., Steghöfer, J.-P., Anders, G., and Reif, W. (2011). Constraining self-organisation through corridors of correct behaviour: The restore invariant approach. In *Organic Computing—A Paradigm Shift for Complex Systems*, pages 79–93. Springer.

Parra-Ullauri, J. M., García-Domínguez, A., García-Paucar, L. H., and Bencomo, N. (2020). Temporal models for history-aware explainability. In *Proceedings of the 12th System Analysis and Modelling Conference*, pages 155–164.

Rudolph, S., Hihn, R., Tomforde, S., and Hähner, J. (2016). Comparison of dependency measures for the detection of mutual influences in organic computing systems. In *Architecture of Computing Systems - ARCS 2016 - 29th International Conference, Nuremberg, Germany, April 4-7, 2016, Proceedings*, pages 334–347.

Rudolph, S., Tomforde, S., and Hähner, J. (2019). Mutual influence-aware runtime learning of self-adaptation behavior. *ACM Trans. Auton. Adapt. Syst.*, 14(1):4:1–4:37.

Schmeck, H., Müller-Schloer, C., Cakar, E., Mnif, M., and Richter, U. (2010). Adaptivity and self-organization in organic computing systems. *ACM Trans. Auton. Adapt. Syst.*, 5(3):10:1–10:32.

Tennenhouse, D. (2000). Proactive computing. *Communications of the ACM*, 43(5):43–50.

Tomforde, S. and Goller, M. (2020). To adapt or not to adapt: A quantification technique for measuring an expected degree of self-adaptation. *Comput.*, 9(1):21.

Tomforde, S., Kantert, J., Müller-Schloer, C., Bödelt, S., and Sick, B. (2018). Comparing the effects of disturbances in self-adaptive systems - A generalised approach for the quantification of robustness. *Trans. Comput. Collect. Intell.*, 28:193–220.

Tomforde, S., Kantert, J., and Sick, B. (2017a). Measuring self-organisation at runtime - A quantification method based on divergence measures. In *Proc. of 9th Int. Conf. on Agents and Art. Int.*, pages 96–106.

Tomforde, S., Prothmann, H., Branke, J., Hähner, J., Mnif, M., Müller-Schloer, C., Richter, U., and Schmeck, H. (2011). Observation and Control of Organic Systems. In Müller-Schloer, C., Schmeck, H., and Ungerer, T., editors, *Organic Computing - A Paradigm Shift for Complex Systems*, Autonomic Systems, pages 325 – 338. Birkhäuser Verlag.

Tomforde, S., Sick, B., and Müller-Schloer, C. (2017b). Organic computing in the spotlight. *CoRR*, abs/1701.08125.

Weber, R. H. and Weber, R. (2010). *Internet of things*, volume 12. Springer.

Weiser, M. (1999). The computer for the 21st century. *ACM SIGMOBILE mobile computing and communications review*, 3(3):3–11.

Welsh, K., Bencomo, N., Sawyer, P., and Whittle, J. (2014). Self-explanation in adaptive systems based on runtime goal-based models. In *Transactions on Computational Collective Intelligence XVI*, pages 122–145. Springer.