Chapter 1
The Notion of Self-Aware Computing

Samuel Kounev, Peter Lewis, Kirstie Bellman, Nelly Bencomo, Javier Camara, Ada Diaconescu, Lukas Esterle, Kurt Geihs, Holger Giese, Sebastian Götz, Paola Inverardi, Jeffrey Kephart and Andrea Zisman

Samuel Kounev
Universität Würzburg, Am Hubland, 97074 Würzburg, Germany e-mail: skounev@acm.org

Peter Lewis
Aston University, Birmingham, B4 7ET, UK e-mail: p.lewis@aston.ac.uk

Kirstie Bellman
The Aerospace Corporation, Los Angeles, US e-mail: Kirstie.L.Bellman@aero.org

Nelly Bencomo
Aston University, B4 7ET, Birmingham, UK e-mail: nelly@acm.org

Javier Camara
Carnegie Mellon University, Pittsburgh, PA 15213 e-mail: jcmoreno@cs.cmu.edu

Ada Diaconescu
Telecom Paris Tech, 75013 Paris, France e-mail: ada.diaconescu@telecom-paristech.fr

Lukas Esterle
Vienna University of Technology, Department of CS, Vienna, Austria e-mail: lukas.esterle@tuwien.ac.at

Kurt Geihs
University of Kassel, 34121 Kassel, Germany e-mail: geihs@uni-kassel.de

Holger Giese
Hasso-Plattner-Institut, 14482 Potsdam, Germany e-mail: Holger.Giese@hpi.de

Sebastian Götz
University of Technology Dresden, Germany e-mail: sebastian.goetz@acm.org

Paola Inverardi
Universit degli Studi dell'Aquila, 67100 L'Aquila, Italy e-mail: paola.inverardi@univaq.it

Jeffrey Kephart
Thomas J. Watson Research Center, NY, USA e-mail: kephart@us.ibm.com

Andrea Zisman
The Open University, UK e-mail: andrea.zisman@open.ac.uk

Abstract We define the notion of "self-aware computing" and the relationship of this term to related terms such as autonomic computing, self-management, and similar. The need for a new definition, driven by trends that are only partially addressed by existing areas of research, is motivated. The semantics of the provided definition are discussed in detail examining the selected wording and explaining its meaning to avoid misleading interpretations. The chapter also provides an overview of existing usage of the terms self-aware computing, respectively self-awareness, in related past projects and initiatives.

1.1 Introduction

There have been a number of research projects and initiatives in computer science and engineering that have explicitly engaged with the notion of self-awareness in computing. Some examples include the SElf-awarE Computing (SEEC) project at MIT and University of Chicago, the ASCENS and EPiCS FP7 EU Projects, the FOCAS FET Coordination Action, and the SEAMS Dagstuhl Seminars and workshop series. Relevant work can be found in several different areas and communities including autonomic computing, machine learning and artificial intelligence, multi-agent systems, self-organizing and self-adaptive systems, situation- and context-aware systems, reflective computing, model-predictive control, as well as work from the models@run-time community. Recent reviews of relevant work [21,29,32] have grouped contributions either by community or thematically, and have found that the terms themselves often lack precise definitions.

This book adopts the notion of *self-aware computing* as defined by the Dagstuhl Seminar 15041 "Model-driven Algorithms and Architectures for Self-Aware Computing Systems"[1] held on January 18-23, 2015. The seminar brought together researchers from the respective communities to discuss past and future trends in self-aware computing and encourage active collaborations and cross-fertilization between related research efforts. An important first step in this direction was the formulation of a new definition of the term "self-aware computing system" integrating the different ways in which this term is used in the interdisciplinary research landscape. In this chapter, we introduce this new broader notion of "self-aware computing" and provide an overview of previous projects and initiatives that have explicitly used this term in the past. By providing a common language, we aim to foster interaction and collaborations between the respective research communities, raising the awareness about related research efforts and synergies that can be exploited to advance the state-of-the-art.

[1] http://www.dagstuhl.de/15041

1.2 Definition of Self-Aware Computing

Definition 1.1. Self-aware computing systems are computing systems that:

1. *learn models* capturing *knowledge* about themselves and their environment (such as their structure, design, state, possible actions, and run-time behavior) on an ongoing basis and
2. *reason* using the models (for example predict, analyze, consider, plan) enabling them to *act* based on their knowledge and reasoning (for example explore, explain, report, suggest, self-adapt, or impact their environment)

in accordance with *higher-level goals*, which may also be subject to change.

It is assumed that a self-aware system is built by an entity with some higher-level goals in mind. This entity may be a human (e.g., a developer) or a set of humans (e.g., a developer team), but it doesn't necessarily have to be. The entity that built the system may also be another computing system, at a higher-level, that generates a new system for a given purpose (e.g., in the form of executable code or models).

The major distinctive characteristics of a self-aware computing system are: i) it must have the capability to learn models on an *ongoing basis*, capturing knowledge relevant to the purpose for which it is built, ii) it must be able to use the models to reason about this knowledge and act accordingly. Both the learning and reasoning part are driven by the system's goals, which may be established by the entity that built the system, by the end user of the system, or by a combination of the two. The goals are referred to as *higher-level* goals to emphasize that they are at a higher-level of abstraction than the system itself and they are not under its direct control. Note that the system itself may generate its own goals (at lower levels) as part of its learning and model-based reasoning processes.

It is assumed that the learned models capture knowledge about the system and its environment. We note that for some systems the boundary between what is considered "knowledge about the system itself" and what is considered "knowledge about the environment" is somewhat blurred. Therefore, we do not require a strict separation of the learned models with respect to whether they capture knowledge about the self or knowledge about the environment in which the self operates. However, it is expected that both the amount and scope of the acquired knowledge should be driven by the higher-level goals and what information could possibly be of use to achieving them.

The term "model" is used here in a general sense and refers to any abstraction of the system and its environment that captures some knowledge and may be used for reasoning with respect to the system goals. In his general model theory, Stachowiak [33] identifies the following three features as essential for models: i) *mapping:* a model is always a model of some *original* (which can be a model itself), ii) *reduction:* a model always *abstracts* from the original by reflecting only a subset of its attributes, and iii) *pragmatic:* a model only replaces the original for a certain *purpose*. We note that "purpose" in this context is understood as potential purpose or a general utility, allowing for exploratory behavior. Usually, we further distinguish *descriptive* models, which capture the originals as they are, from *prescriptive* models, which describe envisioned futures (planned originals). Descriptive models, in our context, describe a given system aspect that may be relevant with respect to the system's higher-level goals. We further distinguish *predictive models* that support more complex reasoning such as, for example, predicting the system behavior under given conditions or predicting the impact of a considered possible adaptation action. We note that the descriptive, prescriptive and predictive model categories are not strictly mutually exclusive as both descriptive and prescriptive models can also be used for predictive analysis and therefore can be predictive models as well.

Some examples of different types of models capturing various aspects that may be relevant in a given scenario include:

- a descriptive model capturing the system's resource landscape and software architecture and their performance-relevant parameters
- a descriptive model describing the system's possible adaptation actions (degrees-of-freedom at run-time)
- a prescriptive model describing how to act in a given situation (e.g., after a component failure)
- a descriptive model describing the system's goals and policies (e.g., service level agreements)
- a predictive statistical regression model capturing the influence of user workloads on the system resource consumption and energy efficiency
- a predictive stochastic model allowing to predict the system performance for a given user workload and resource allocation
- a control theory model used to guide the system behavior

We stress that the term "learn" does imply that some information based on which models are derived is obtained at system run-time, while also additional static information built into the system at design-time can be employed as well. Typically, a combination of both would be expected, for example, a system may be built with integrated skeleton models whose parameters are estimated using monitoring data collected at run-time. The model learning is expected to happen on an *ongoing basis* during operation meaning that models should be continuously refined and calibrated in order to better fulfill the purpose for which they are used.

Taken together the learning and reasoning are expected to enable model-based analysis at run-time that goes beyond applying simple rules or heuristics explicitly programmed at system design-time. Depending on the considered type of system and its respective goals, different types of model-based reasoning may be relevant. For example, in the context of an IT system that has been designed to guarantee certain performance requirements, the following types of reasoning may be relevant:

- predict the load of an IT system (e.g., number of users or requests sent per unit of time) in a future time horizon

- predict the system performance (e.g., response time) for a given workload and resource allocation (e.g., number of servers)
- predict the expected impact of a given system adaptation action (e.g., adding or removing system resources) on the end-to-end system performance
- determine how much resources need to be added to ensure that performance requirements are satisfied under an increasing system workload
- estimate the system's energy consumption at runtime and compare it to other system configurations (e.g., with respect to voltage and frequency) in order to select an optimal configuration

An example of reasoning in the context of a cyber-physical system for traffic management may be to analyze the traffic situation in order to provide a recommendation which route to take for a given target destination. Another example, in the context of medical implants, is a system that monitors concentrations of substances in the body to determine if release of medication is needed in case these substances are not at the right level. Further, a system that monitors signals in the body (those before an epileptic attack) in order to issue a warning signal / alarm in case patterns are identified that match the pattern of an attack.

To clarify our intention, we compare the terms *learning*, *reasoning* and *acting*, employed in our definition, with the weaker alternative terms *observing* and *reacting*. While *observing* is a critical prerequisite for learning, *learning* clearly goes beyond only observing by accumulating knowledge about the subject of observation over time and manifesting this knowledge in reflective models. Also *reacting* can be the result of *reasoning* and *acting*, however, a system that only reacts, but does not reason, does not have the capability to consider the current situation and its options before it takes any action. Therefore, according to our definition, a self-aware computing system is expected to not only observe and react, but also to learn, reason, and act.

By stressing the role of model learning and model-based reasoning, driven by higher-level goals, we distinguish the term self-aware computing from related terms such as autonomic computing or self-adaptive systems. Although, in most cases, it would be expected that the system uses the learned models to reason *and self-adapt* to changes in the environment, we note that self-adaptation (often referred to as *self-expression* in this context) is not strictly required. In this way, we accommodate cases where all adaptation actions must be supervised and authorized by an entity outside of the system, such as the entity that built the system or a human system user. For example, in mission-critical cognitive computing applications, systems may provide recommendations on how to act, however, the final decision on what specific action to take is often made by a human operator.

1.3 Previous Initiatives in Self-Aware Computing

As mentioned earlier, there have been a number of research projects and initiatives that have explicitly engaged with the notion of *self-awareness* in computing. Here, we provide an overview of some of the most significant efforts, perspectives and key contributions. In the next chapter, we discuss previous work in self-aware computing in more detail, contextualising the perspectives taken in this book.

1.3.1 Self-awareness in Artificial Intelligence

Self-awareness has long been of interest in the artificial intelligence (AI) community. In particular, studies have focused on higher levels of self-awareness such as *meta-self-awareness*: a system's awareness of its own self-awareness. This concept overlaps significantly with meta-cognition, defined by Metcalfe and Shimamura [22] as *knowing about knowing*. Integration of AI technologies into systems that, as a result of the integration, exhibit self-awareness of this meta-cognitive form, has been on DARPA's research agenda for some time [27]. Indeed, architectural issues in building such integrated systems which then exhibit self-awareness were the subject of a DARPA workshop [4] in 2004.

One example of the consideration of self-awareness in the artificial intelligence community is the algorithm selection problem, where a system can reason about its own reasoning, that is, it possesses meta-cognition, and can select an appropriate reasoning method according to its situation [8]. In autonomous robotics, it has been argued that self-awareness is not only beneficial, but essential for safety and robot ethics [29].

Furthermore, self-awareness is not only a property that can be observed at an individual level, but also something that can arise in a collective intelligence context. For example, a group of robots with simple behavioural rules and local interactions may arrive at an emergent awareness of the properties of the group, including for example its history and interactions with the world, though this awareness is distributed across the individual units [23].

1.3.2 Engineering Self-Aware Systems

While meta-cognition or meta-self-awareness are concerned with higher reasoning abilities, and are of particular interest in artificial intelligence, efforts exist at a more fundamental level to engineer systems that explicitly consider knowledge about themselves. Agarwal et al. [1, 2] put forward a case for a paradigm shift in system design practice. The idea here is to move from a procedural design methodology wherein the behaviour of the computing system is pre-programmed or considered beforehand (i.e., at design time), towards a self-aware system where this is not required and the system adapts to its context at run-time. One aim is to avoid or reduce the need to consider the availability of resources and various other constraints beforehand, instead intelligently trading-off available resources for performance at run-time.

For example, the programming effort required to build a system to satisfy carefully considered specifications could be reduced if there are resources at hand for the system to use in an automated manner in order to optimise its own behaviour, such that it achieves a goal given current constraints. A self-aware computer is thus given a goal and it automatically works to achieve the goal, for example with the minimal amount of energy or other resources. In order to facilitate the engineering of such systems, Hoffman et al. have developed a general and extensible framework for self-aware computing [14], which integrates both control theory and machine learning methods.

Self-awareness has also been proposed as a way to tackle the increasing complexity and dynamics associated with modern service-based IT systems. Kounev [19] proposes that self-aware software services, which have built-in models of their own architecture and dynamic aspects of the system's interactions with its environment, can improve the utilisation of resources, while continuing to satisfy quality of service requirements. Further, Kounev et al. [18] highlight self-reflection, self-prediction and self-adaptation as key characteristics of self-aware service-based systems, and propose that methodologies for the systematic engineering of self-aware systems are needed.

Importantly, for a system to be self-aware it is not required to be highly complex; indeed the scalability of the concept means that self-awareness has also been considered in much simpler systems. An example of this is so called *cognitive radio devices* [11], which monitor and control their own capabilities and also communicate with other radio devices to monitor theirs. This enables them to improve the efficiency of communication by negotiating changes in parameter settings [34]. We will explore a wide range of self-awareness concepts in the following chapter, where both minimal examples such as this, and more "full stack" self-aware systems will be compared.

In order to design self-aware systems, various enabling technologies have been proposed in the form of application heartbeats [13, 31], for establishing a standard way of setting application goals and evaluating the performance of the system whilst trying to achieve the goals, and the use of heuristic and machine learning techniques for adaptation and decision making on the part of the system. An example of the latter is the use of reinforcement learning in order to optimise a scheduling policy for system components to gain access to the critical section within applications [10].

1.3.3 Self-Awareness in Pervasive Computing

The Pervasive Computing community is also interested in self-awareness, due to the typically mobile characteristics of agents, whose context continuously changes. As such, they need to synthesise knowledge based on their own state and their environment, in order to adapt to changes. Often monitoring and adaptation are studied in the context of human-computer interaction, since the interest is on how such systems self-adapt in order to be useful to humans in different situations (e.g., "going for a run”). Ye et al. [35] discuss issues and challenges involved in assimilating sensor data from a myriad of sources in order for pervasive computing systems to identify situations that human users may be in. They show a shift in techniques over time from *logic based* ones towards those that are *learning based*, as the sensor data has become more complex, erroneous and uncertain, with sensors becoming ever more pervasive. The learning of mappings between sensor data and the situation, given current model building techniques, poses challenges such as the lack of training data, which can lead to low performing models. This has been tackled by considering unsupervised learning [5,12] and web mining [28], allowing for extracting common sense knowledge. Another line of research within pervasive computing concerns constructing simulation models of contexts, for applications to be tested in [15].

1.3.4 Systems with Decentralised Self-Awareness

Self-awareness research is not limited to an entity or system in itself being able to monitor and reason about itself, but also describes emergent phenomena [23] in collective systems. In natural systems like ant colonies and the immune system, the awareness of the global state is distributed across the elementary units that make up the system (e.g., ants and their trails) and is statistical in nature. This helps the system stay robust at the global level in the face of disturbances. In essence, the system as a whole is aware enough of itself to understand when the globally stable state gets disturbed, and engages the elementary units to collect information locally, which builds up in a statistical fashion, helping the elementary units use this statistical information to get the system back into the globally stable state. Mitchell proposed [23] that such systems can provide guidelines for designing artificial intelligence systems with decentralised architectures, for example robotic swarms, which exhibit apparent self-awareness.

One example of where such a system has been developed is within the SWARM-BOTS project [24]. One of the objectives of the SWARM-BOTS project was the design and implementation of a novel mobile robot, called an *s-bot*. While s-bots’ individual capabilities within an environment are physically limited (much like individual insects in the natural world), through local communication they are able to *self-assemble* [9] into larger structures, known as *swarm-bots*, which are capable of achieving goals not reachable by individual s-bots. Examples of such goals might include navigation over challenging terrain or the transportation of large objects; in all cases, these tasks cannot be solved without the coordinated movements of individual *s-bots*.

In this context, it is important to consider the relationship between individual systems’ self-awareness, the self-awareness of neighbouring systems, and the self-awareness of the collective as a whole. Zambonelli et al. [36] discuss some of the issues involved here, considering aspects of self-awareness relating to the ability of components to recognise situations, and changes in situations (both internal and external) in a collective system. This self-awareness is then used to drive self-expression, including adaptation to the new situation. Importantly, this self-expression also occurs at the level of the collective.

Other research challenges within the area of self-assembly, or *structural self-organisation*, include better understanding how system structures, rather than individual behaviours, can be adapted over time with respect to the system's distributed sense of *self-concept* [17].

1.3.5 Computational Self-Awareness

Lewis et al. [20] propose that human self-awareness can serve as a source of inspiration for a new notion of *computational self-awareness*, and associated *self-expression*, behaviour based on self-awareness. These comprise various capabilities, encapsulated as different "levels" of self-awareness, based on cognitive psychologist Ulric Neisser's [25] broad set of levels of human self-awareness. The intention is to explicitly account for a full spectrum of existing and future systems, including simple systems which sense and learn about themselves, as well as what might typically be considered highly advanced artificial intelligence. The levels of computational self-awareness provide one axis on which to compare *how* self-aware a computing system is. Lewis et al. argue that increased self-awareness (in terms of the levels) can improve a system's ability to manage complex trade-offs in changing conditions.

They introduce a general framework for the description of the self-awareness properties of computing systems, which includes a reference architecture and a series of derived architectural patterns. These can be used by engineers to determine whether, how, and to what extent to build self-awareness capabilities into a system. The framework proposed by Lewis et al. has been used to consider the self-awareness properties of such diverse systems as distributed smart cameras [30], heterogeneous reconfigurable multi-core systems [3], and fault tolerance in avionic systems [26].

Lewis et al.'s notion of *computational self-awareness* [20] intentionally includes many existing and prior systems, which have not been previously described as self-aware. Nevertheless these systems use capabilities, which fulfil some self-awareness aspect, often due to their benefit in a complex environment.

1.4 A Concept of a Self-Aware Learning and Reasoning Loop

In autonomic computing, the system behavior is typically represented as a control loop. To structure the principle of operation exhibited by autonomic managers, [16] defined a reference architecture based on a control loop, typically referred to as the MAPE-K loop. This reference architecture has the advantage that it offers a clear way to identify and classify areas of particular focus and thus, it is used by many researchers to communicate the architectural concepts of autonomic systems.

In the following, we contrast the MAPE-K loop used in autonomic computing to the concept of self-aware computing as defined above. To this end, we first briefly describe the MAPE-K loop. The acronym MAPE-K reflects the five main constituent phases of autonomic loops, i.e., MONITOR, ANALYZE, PLAN, EXECUTE, and KNOWLEDGE, as depicted in Figure 1.1. Basically, the MONITOR phase collects information from the sensors provided by the managed artifacts and its context. The ANALYZE phase uses the data of the MONITOR phase to assess the situation and determine any anomalies or problems. The PLAN phase generates an adaptation plan to solve a detected problem. The EXECUTE phase finally applies the generated adaptation plan on the actual system. A cross-cutting aspect shared among all phases of the loop is the KNOWLEDGE about the system and its context, capturing aspects like the software architecture, execution environment, and hardware infrastructure on which the system is running.

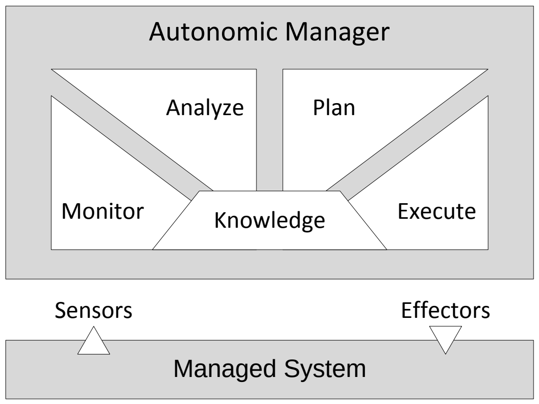

Fig. 1.1: MAPE-K autonomic control loop [16].

The software engineering community uses a similar feedback loop concept, distinguishing the four phases COLLECT, ANALYZE, DECIDE, and ACT [7]. Conceptually, the behavior of these phases is similar to the phases in the MAPE-K loop, however, this concept does not explicitly consider the KNOWLEDGE part. More details about the use of feedback loops in self-adaptive systems, such as the use of multiple, multi-level, positive, or negative feedback loops, are given by [6].

Although the notion of self-aware computing, as defined in this book, has some common aspects with the concept of a feedback loop, such as MAPE-K, there are some important differences. While the MONITOR and ANALYZE phases of MAPE-K imply that information is gathered and analyzed continually at run-time, this does not necessarily imply that the acquired information is abstracted and used to learn models on an ongoing basis during operation. Similarly, while taken together the ANALYZE and PLAN phases can be compared to the model-based learning and reasoning process in self-aware computing, the latter requires that the type of analysis conducted at run-time based on the learned models goes beyond applying simple rules or heuristics explicitly programmed at system design-time. Finally, the EXECUTE phase in MAPE-K implies that the system self-adapts at run-time. In contrast, as mentioned earlier, self-adaptation (or self-expression) is not strictly required in self-aware computing. A self-aware computing system may provide recommendations on how to act, however, the final decision on what specific action to take may be left to a human operator, for example, as typical for many cognitive computing applications.

Figure 1.2 illustrates our concept of a model-based learning and reasoning loop (LRA-M loop) capturing the main activities in a self-aware computing system. The figure shows the *self* and its interfaces to the environment in which it is operating. The activities within the self are driven by its goals and its observations collected as empirical data about relevant aspects of the system and its environment, its users, and so on. The empirical data is used as a basis for the ongoing LEARNING process, as part of which observations are abstracted into models capturing potentially relevant aspects of the system and its environment (such as their structure, design, state, possible actions, and run-time behavior). The learned models form the system's knowledge base (corresponding to the KNOWLEDGE part in the MAPE-K loop), which provides the basis for the system's REASONING process. The reasoning process may trigger ACTIONS affecting both the behavior of the system itself (self-adaptation) as well as possibly impacting the environment. The actions may also affect the system's learning and reasoning activities themselves, for example, by focussing the learning process on selected aspects or observations. We note here that although we explicitly distinguish the learning process from the reasoning process, it is not strictly required that in a self-aware computing system these processes are separated, since in many cases the two activities may be interweaved.

1.5 Conclusion

The two major distinctive characteristics of a self-aware computing system are: i) it must have the capability to learn models on an *ongoing basis*, capturing knowledge relevant to the purpose for which it is built, ii) it must be able to use the models to reason about this knowledge and act accordingly. The term "model" refers to any abstraction of the system and its environment that captures some knowledge and may be used for reasoning with respect to the system goals. Both the learning and reasoning part are driven by the system's goals, which may be established by the entity that built the system, by the end user of the system, or by a combination of the two. Taken together the learning and reasoning are expected to enable model-based analysis at run-time that goes beyond applying simple rules or heuristics explicitly programmed at system design-time. As discussed in this introductory chapter, the notion of "self-aware computing" is strongly related to existing notions like auto-

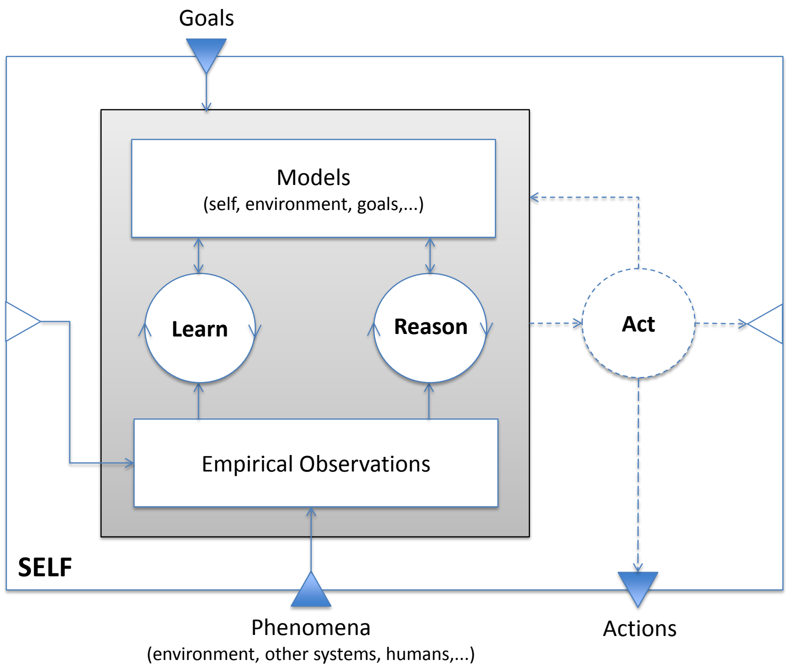
Goals
Models
(self, environment, goals,...)
Learn
Reason
Act
Empirical Observations
SELF
Phenomena
(environment, other systems, humans,...)
Actions

Fig. 1.2: Self-aware learning and reasoning loop: LRA-M loop.

nomic computing, self-management, and similar. The novelty of the term is that it explicitly stresses model learning and reasoning as ongoing processes built into a system's design. Thus, the role of models capturing static and dynamic knowledge about the system, as well as the use of model-driven algorithms and architectures as a basis for reasoning, are central to the vision of self-aware computing systems.

References

1. Anant Agarwal and Bill Harrod. Organic computing. Technical Report White paper, MIT and DARPA, 2006.
2. Anant Agarwal, Jason Miller, Jonathan Eastep, David Wentziaff, and Harshad Kasture. Self-aware computing. Technical Report AFRL-RI-RS-TR-2009-161, MIT, 2009.
3. Andreas Agne, Markus Happe, Achim Lsch, Christian Plessl, and Marco Platzner. Self-awareness as a model for designing and operating heterogeneous multicores. *ACM Trans. on Reconfigurable Technology and Systems (TRETS)*, 7(2):13:11–13:18, 2014.
4. Eyal Amir, M. L. Anderson, and Vinay K. Chaudhri. Report on darpa workshop on self-aware computer systems. Technical Report UIUCDCS-R-2007-2810, UIUC Comp. Sci., 2007.
5. Oliver Brdiczka, James L. Crowley, and Patrick Reignier. Learning situation models in a smart home. *IEEE Trans. Sys. Man Cyber. Part B*, 39:56–63, 2009.
6. Yuriy Brun, Giovanna Marzo Serugendo, Cristina Gacek, Holger Giese, Holger Kienle, Marin Litoiu, Hausi Müller, Mauro Pezzè, and Mary Shaw. Engineering self-adaptive systems

through feedback loops. In Betty H. Cheng, Rogério Lemos, Holger Giese, Paola Inverardi, and Jeff Magee, editors, *Software Engineering for Self-Adaptive Systems*, pages 48–70. Springer-Verlag, Berlin, Heidelberg, 2009.

7. Betty H. C. Cheng, Rogério Lemos, Holger Giese, Paola Inverardi, Jeff Magee, Jesper Andersson, Basil Becker, Nelly Bencomo, Yuriy Brun, Bojan Cukic, Giovanna Marzo Serugendo, Schahram Dustdar, Anthony Finkelstein, Cristina Gacek, Kurt Geihs, Vincenzo Grassi, Gabor Karsai, HolgerM. Kienle, Jeff Kramer, Marin Litoiu, Sam Malek, Raffaela Mirandola, HausiA. Müller, Sooyong Park, Mary Shaw, Matthias Tichy, Massimo Tivoli, Danny Weyns, and Jon Whittle. Software Engineering for Self-Adaptive Systems: A Research Roadmap. In Betty H. C. Cheng, Rogério Lemos, Holger Giese, Paola Inverardi, and Jeff Magee, editors, *Software Engineering for Self-Adaptive Systems*, volume 5525 of *Lecture Notes in Computer Science*, pages 1–26. Springer Berlin Heidelberg, 2009.
8. M.T. Cox. Metacognition in computation: A selected research review. *Art. Int.*, 169(2):104–141, 2005.
9. Marco Dorigo, Vito Trianni, Erol Şahin, Roderich Groß, Thomas H. Labella, Gianluca Baldassarre, Stefano Nolfi, Jean-Louis Deneubourg, Francesco Mondada, Dario Floreano, and Luca M. Gambardella. Evolving self-organizing behaviors for a swarm-bot. *Autonomous Robots*, 17:223–245, 2004.
10. J. Eastep, D. Wingate, M.D. Santambrogio, and A. Agarwal. Smartlocks: lock acquisition scheduling for self-aware synchronization. In *Proceeding of the 7th international conference on Autonomic computing*, pages 215–224. ACM, 2010.
11. B.A. Fette. *Cognitive radio technology*. Academic Press, 2009.
12. Raffay Hamid, Siddhartha Maddi, Amos Johnson, Aaron Bobick, Irfan Essa, and Charles Isbell. A novel sequence representation for unsupervised analysis of human activities. *Artificial Intelligence*, 173(14):1221 – 1244, 2009.
13. H. Hoffmann, J. Eastep, M.D. Santambrogio, J.E. Miller, and A. Agarwal. Application heartbeats for software performance and health. In *ACM SIGPLAN Notices*, volume 45, pages 347–348. ACM, 2010.
14. Henry Hoffmann, Martina Maggio, Marco D. Santambrogio, Alberto Leva, , and Anant Agarwal. Seec: A general and extensible framework for self-aware computing. Technical Report MIT-CSAIL-TR-2011-046, MIT CSAIL, 2011.
15. M.C. Huebscher and J.A. McCann. Simulation model for self-adaptive applications in pervasive computing. In *Proceedings of the Database and Expert Systems Applications, 15th International Workshop*, pages 694–698. IEEE Computer Society, 2004.
16. Jeffrey O. Kephart and David M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.
17. Serge Kernbach. From robot swarm to artificial organisms: Self-organization of structures, adaptivity and self-development. In Paul Levi and Serge Kernbach, editors, *Symbiotic Multi-Robot Organisms*, volume 7. Springer, 2010.
18. Samuel Kounev. Engineering of Self-Aware IT Systems and Services: State-of-the-Art and Research Challenges. In *Proceedings of the 8th European Performance Engineering Workshop (EPEW'11), Borrowdale, The English Lake District, October 12–13*, 2011. (Keynote Talk).
19. Samuel Kounev. Self-Aware Software and Systems Engineering: A Vision and Research Roadmap. In *GI Softwaretechnik-Trends, 31(4), November 2011, ISSN 0720-8928*, Karlsruhe, Germany, 2011.
20. Peter R. Lewis, Arjun Chandra, Funmilade Faniyi, Kyrre Glette, Tao Chen, Rami Bahsoon, Jim Torresen, and Xin Yao. Architectural aspects of self-aware and self-expressive computing systems. *IEEE Computer*, 2015.
21. Peter R. Lewis, Arjun Chandra, Shaun Parsons, Edward Robinson, Kyrre Glette, Rami Bahsoon, Jim Torresen, and Xin Yao. A survey of self-awareness and its application in computing systems. In *Proc. Int. Conference on Self-Adaptive and Self-Organizing Systems Workshops (SASOW)*, pages 102–107, Ann Arbor, MI, USA, 2011. IEEE Computer Society.
22. Janet Metcalfe and Arthur P. Shimamura, editors. *Metacognition: Knowing about knowing*. MIT Press, Cambridge, MA, USA, 1994.

23. Melanie Mitchell. Self-awareness and control in decentralized systems (Tech Report SS-05-04). In *AAAI Spring Symposium on Metacognition in Computation*, Menlo Park, 2005. AIII Press.
24. Francesco Mondada, Giovanni C. Pettinaro, Andre Guignard, Ivo W. Kwee, Dario Floreano, Jean-Louis Deneubourg, Stefano Nolfi, Luca Maria Gambardella, and Marco Dorigo. Swarm-bot: A new distributed robotic concept. *Autonomous Robots*, 17:193–221, 2004.
25. Ulric Neisser. The roots of self-knowledge: Perceiving self, it, and thou. *Annals of the New York Academy of Sciences*, 818:19–33, 1997.
26. Tatiana Djaba Nya, Stephan C. Stilkerich, and Cristian Seimers. Self-aware and self-expressive driven fault tolerance for embedded systems. In *IEEE Symposium on Intelligent Embedded Systems (IES)*, pages 27–33. IEEE, 2014.
27. L.D. Paulson. DARPA creating self-aware computing. *Computer*, 36(3):24, 2003.
28. Mike Perkowitz, Matthai Philipose, Kenneth Fishkin, and Donald J. Patterson. Mining models of human activities from the web. In *Proceedings of the 13th international conference on World Wide Web*, pages 573–582, 2004.
29. Jeremy Pitt, editor. *The Computer After Me: Awareness and Self-awareness in Autonomic Systems*. Imperial College Press, 2014.
30. Bernhard Rinner, Lukas Esterle, Jennifer Simonjan, Georg Nebehay, Roman Pflugfelder, Peter R. Lewis, and Gustavo Fernndez Domnguez. Self-aware and self-expressive camera networks. *IEEE Computer*, 2015.
31. M.D. Santambrogio, H. Hoffmann, J. Eastep, and A. Agarwal. Enabling technologies for self-aware adaptive systems. In *Adaptive Hardware and Systems (AHS), 2010 NASA/ESA Conference on*, pages 149–156. IEEE, 2010.
32. J. Schaumeier, J. Pitt, and G. Cabri. A tripartite analytic framework for characterising awareness and self-awareness in autonomic systems research. In *Self-Adaptive and Self-Organizing Systems Workshops (SASOW), 2012 Sixth IEEE Conference on*, pages 157–162, 2012.
33. Herbert Stachowiak. *Allgemeine Modelltheorie*. Springer, Wien, 1973.
34. J. Wang, D. Brady, K. Baclawski, M. Kokar, and L. Lechowicz. The use of ontologies for the self-awareness of the communication nodes. In *Proceedings of the Software Defined Radio Technical Conference SDR*, volume 3, 2003.
35. Juan Ye, Simon Dobson, and Susan McKeever. Situation identification techniques in pervasive computing: A review. *Pervasive and Mobile Computing*, In Press., 2011.
36. F. Zambonelli, N. Bicocchi, G. Cabri, L. Leonardi, and M. Puviani. On self-adaptation, self-expression, and self-awareness in autonomic service component ensembles. In *Self-Adaptive and Self-Organizing Systems Workshops (SASOW), 2011 Fifth IEEE Conference on*, pages 108–113, 2011.