“Know thyself” – Computational Self-Reflection in Intelligent Technical Systems

Sven Tomforde, Jörg Hähner, Sebastian von Mammen
University of Augsburg
{sven.tomforde|joerg.haehner|sebastian.von.mammen}
@informatik.uni-augsburg.de

Christian Gruhl, Bernhard Sick, Kurt Geihs
University of Kassel
{cgruhl|bsick|geihs}
@uni-kassel.de

***Abstract*—In many application domains, developers aim at building technical systems that can cope with the complexity of the world they are surrounded with, including other technical systems. Due to this complexity, system designers cannot explicitly foresee every possible situation “their” system will be confronted with at runtime. This resulted in solutions capable of self-adaptation at runtime. Future intelligent technical systems will have to go far beyond such a reactive solution – the general question is: How can systems themselves define new goals and new classes of goals in order to increase their own performance at runtime and without the need of human control or supervision? This paper introduces a definition of “computational self-reflection”, proposes an architectural concept, and discusses the potential benefit by means of three exemplary application scenarios. Finally, building blocks to achieve self-reflection are discussed and a basic research agenda is drafted.**

I. Motivation

ΓΝΩΘΙ ΣΑΥΤΟΝ (in English: *Know thyself*) is said to have been inscribed at the wall of the Apollo’s temple at Delphi in ancient times [1]. This Greek aphorism is often regarded as a basic directive for human behavior [2].

Self-knowledge basically refers to knowledge of one’s mental state, which includes beliefs, desires, and sensations [3]. It also includes knowledge about own character traits, manual, mental, and social skills, strengths, weaknesses, or limitations [4]. Essentially, “know thyself” expresses the need to know what we know (and what we can afford) and what we do not know (and what we better should not try to achieve) [2]. Self-knowledge is complemented by knowledge about the “external” world [3]. Self-knowledge is a term that not only applies to individuals, but also to groups of individuals or human societies [2].

Self-reflection is closely related to the concept of self-knowledge. In general, it aims at exploring (and, of course, at exploiting) the possibilities and conditions to gather experience [5]. Thus, humans gain some kind of “knowledge about knowledge” by “thinking about thinking” [6]. Human self-reflection is based on the capacity of humans for introspection in order to learn more about one’s “self” [7]. Today, the term “self-reflection” is often supplanted by the term “introspection” [8].

Introspection, which basically means “looking within” (self-observation or self-monitoring [9]), refers to the ability of humans to learn about their own (current or past) mental states (see above) or processes [10]. Such, introspection is a key concept in epistemology [10]. Introspection complements other sources of knowledge such as perception, memory, or testimony [11].

We claim that a computerized technical system needs some kind of self-reflection to act and react intelligently. As a key contribution of this article, we introduce and discuss a holistic concept of self-reflection for such systems, which includes techniques for self-monitoring, self-assessment, self-awareness, context-awareness, planning, and, ultimately, self-improvement. In our notion of self-reflection, knowledge about the “internal” world must be intimately fused with knowledge about the “external” world. As these self-reflecting systems will be able not only to maintain a certain behavioral level, but even to improve their behavior in uncertain, time-variant environments, we call them “intelligent” (or “smart”). Moreover, if these systems are part of a larger, distributed system of systems (SoS), they will also be able to efficiently and effectively contribute to performance improvements of the overall system. Possible application areas of such systems or SOS are, therefore, smart cities, smart mobility, intelligent robots, cyber-physical systems, or autonomous driving.

Having now introduced some basic terms from the viewpoint of philosophy of mind, this article continues by discussing related work on self-reflection and introspection (Section II). Here, we consider social sciences as well as computer science, where we can find some related work in the field of reflective programming and multi-agent systems, for instance. Next, we define the term self-reflection in the context of intelligent technical systems and present a generic architecture for such systems (Section III). Section IV discusses some use cases in order to identify future research challenges. Possible contributions from related fields to derive solution perspectives for self-reflective systems are identified in Section V, and Section VI combines the key ideas in a conceptual research roadmap that names the basic questions.

II. Related Work

A. Research in Social Sciences

Social sciences make a step from a behavioral to an operational notion of self-reflection. *Reflective practice* is defined as the ability of individuals to reflect systematically on actions in order to learn continuously [12]. Reflection techniques are applied in many areas such as education, healthcare, or management.

Typically, reflective practice is based on some kind of model that describes the various steps that are passed trough.

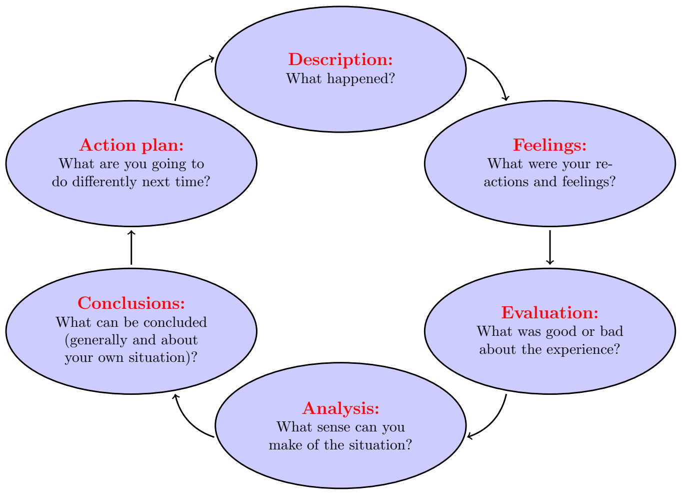
Description:
What happened?
Feelings:
What were your reactions and feelings?
Evaluation:
What was good or bad about the experience?
Analysis:
What sense can you make of the situation?
Conclusions:
What can be concluded (generally and about your own situation)?
Action plan:
What are you going to do differently next time?

Fig. 1. Structured debriefing model defined by Gibbs [13].

Kolb's model for *experiential learning* [14], for instance, encompasses the four steps active experimentation, concrete experience, reflective observation, and abstract conceptualization in a learning cycle.

Another important model was proposed by Gibbs [13] who introduced the concept of *structured debriefing* shown in Fig. 1. Starting with a description of observations (top), the debriefing model encompasses six steps of monitoring and assessing the own behavior as well as deriving conclusions and making new action plans.

Argyris and Schön mention two nested loops in their learning model [15]: An inner loop where individuals basically rely on existing techniques or methods if a situation reoccurs again, even if an error and/or a correction was made, and an outer loop where individuals modify their techniques, methods, or even their objectives to continuously improve. This may be seen as a distinction between learning as a spontaneous act (inner loop) and learning as an explicit process (outer loop).

The development of learning over time is illustrated in Cowan's model shown in Fig. 2 [16]: *Reflection-for-action*, *reflection-in-action*, and *reflection-on-action* describe the temporal relationship between acting and reflecting (anticipatory, simultaneous, and retrospective).

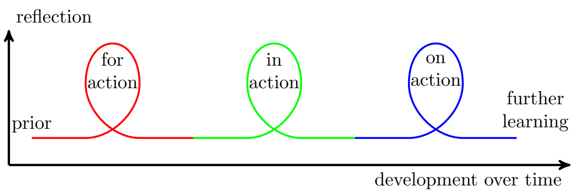

Fig. 2. Illustration of Cowan's learning model [16].

In general, reflection may be founded on many knowledge sources, called "lenses" in the work of Brookfield [17]: the lens of the autobiography of the learner, the lens of the learner's own eyes, the lens of the colleagues' experiences, and finally the lens of the theoretical literature.

B. Research in Computer Science

In the field of computer science, the term *reflection* is historically associated with programming languages and their means of self-modifying programs. Initial research on reflection in programming languages was done in the early 80's by Smith for procedural languages [18] and later extended to object oriented languages by Maes [19]. Maes also defined *computational reflection* as "the activity performed by a computational system when doing computation about (and by that possibly affecting) its own computation." [19]. In some programming systems (e.g., LISP) reflection is implemented using metaprogramming. A metaprogram is basically not different from any other program. It processes data and is certainly able to analyze and also modify it with the virtue that the data it reasons about is again a program. In cases where the metaprogram reasons about itself, which matches the idea of a self-modifying program, the program may be termed to be (self-)reflective. Programming systems that support this type of reflection are called *procedurally reflective*.

Another way to implement reflection is *declarative reflection* where self-representation is not implemented in the system itself, but (as in JAVA, for instance) as an appropriate API. In this domain, the concept of reflection is subdivided into two parts: *introspection* and *intercession*. Introspection essentially matches the definition we gave in Section I ("looking within") and is associated with data analyzing (e.g., what methods belong to a certain class) whereas intercession, i.e., the ability to alter itself, corresponds to data modification. With respect to intercession we further distinguish structural reflection (e.g., adding a new instance variable to a class at runtime) and computational reflection (also behavioral reflection; e.g., altering the code of a method at runtime) [20]. However, the main purpose of reflection in the programming context is not to write self-modifying algorithms but to circumvent restrictions of the programming language or to investigate the structures of objects at runtime (e.g., to check if a certain method is present or for debugging purposes).

Research on reflective systems was also done in the domain of multi-agent systems. Rehák et al. suggest an abstract architecture to enhance multi-agent systems with reflective properties [21]. In essence, the approach is based on the architecture illustrated in Fig. 3. The *reasoning layer* implements the agent's problem solving abilities, while the *reflective layer* lies on top and observes the lower one. Again the process of reflection is subdivided into two parts. The *cognition module*, which implements *self-* and *mutual-awareness*, closely resembles introspection. It manages a model of the agent, the environment, and its social neighborhood. This model is used as foundation for meta-reasoning, that is, to identify what behaviors might be modified to better adapt the system to its environment. The adaptation is done in the *reflection module* where an abstract representation of the desired behavior is processed. This does not necessarily result in new algorithms but in changing parameters of the engaged algorithms to alter their behavior. Therefore, this type of reflection is called *behavioral reflection* opposed to *structural reflection* where internal data structures are reorganized. The authors of [21] also suggest computational reflection itself as technique to manage reflective processes, whereby preceding changes made to the system are used as knowledge base to reflect on.

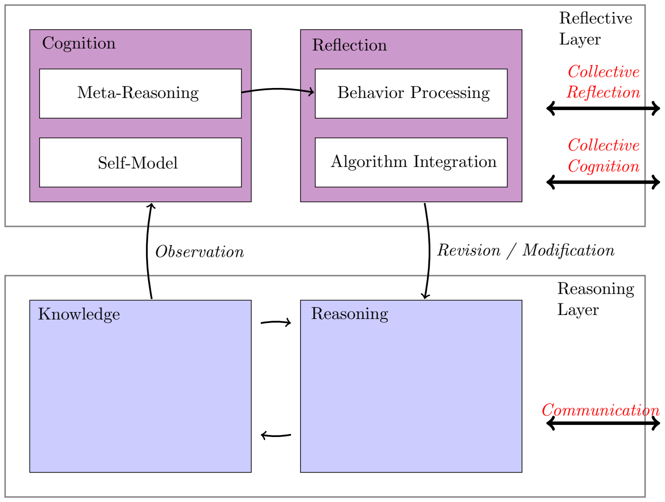

Fig. 3. Architecture of a reflective agent, based on [21].

Bellman and Landauer adopted the concept of computational reflection to solve multi-objective optimization problems [22]. They point out that optimization in dynamic systems is hard, especially in complex systems consisting of multiple layers with different self-optimizing processes. This is due to contradictory objectives and the fact that most self-optimizing processes are not using a self-model in addition to a context model and, thus, are "mindless" (i.e., utilizing single loop learning). To address these problems the different optimization tasks are folded into a web of *reflective processes*, the "reflective web". Basically, the reflective web is a provisional approach until there are mathematical methods that map optimizations of one layer to the next higher layer to combine optimizations. Unfortunately, the organization of the web is not *self-developed* (i.e., autonomously done by the system) and must be done by hand for each application.

To model *reflective processes* so-called *wrappings* are used which comply to a more abstract representation of resources that allows to keep problems and their solutions separated [23]. This leads to a system that does not "call functions", "issue commands", or "send messages" but instead "poses problems". To solve a posed problem, the system "applies" resources (i.e., optimization algorithms) in an automated manner. As a result of representing everything as resources, that can also be replaced at any given time, the system is highly flexible and well fitted to support computational reflection. The described technique was evaluated in the CARS case study, where different cars with diverse properties assessed behavioral strategies under constraints to reach a given goal. The constraints (e.g., low energy consumption or optimal sensor utilization) could be contradictory and even change at runtime, thus creating a dynamic multi-optimization problem. Initially, the characteristics of each car were unknown and an appropriate self-model had to be learned by every system. The evaluation itself relied on computational reflection with different types of reflection (i.e., different scopes of responsibilities) and revealed that computational reflection actually is a possible approach to solve multi-optimization problems – but in this case with the drawback of low autonomy (the reflective web must be organized by hand).

III. Computational Self-Reflection in Intelligent Technical Systems

In this section we give a definition of the term self-reflection in the context of intelligent technical systems and we propose a generic architecture for a system with the ability of self-reflection.

A. A Definition of Computational Self-Reflection

In this article, we will rely on the following working definition of computational self-reflection:

Self-reflection in intelligent technical systems (or computational self-reflection) is the ability of the system to continuously monitor and improve its own behavior in an uncertain, dynamic, and time-invariant environment for situations that may not have been anticipated at design-time of the system.

Self-reflection is based on complementary techniques

1) to monitor and to assess the environment (including other, similar systems) and the system's own behavior,
2) to model the system's own knowledge about the environment and about its own behavior (including meta-knowledge such as experience gained by applying knowledge), and
3) to define own goals, to find new ways to solve these goals (including combination of several goals or conflict solving), and to trigger appropriate changes of own system parameters, algorithms, structure, etc.

These aspects can also be seen as requirements that must be fulfilled to call a system "self-reflective".

Computational self-reflection goes far beyond self-adaptation which could also be seen as the "inner" of two nested loops which are not necessarily temporally synchronized. Self-reflection in the above sense includes (cf. Section II) introspection and intercession, it includes reflection for, in, and on action, and its three basic steps may be refined further and realized as proposed by Gibbs in his structured debriefing model, for instance. Computational self-reflection is based on different knowledge sources such as a system's own sensor information, its experience (and, possibly, prior information provided at design-time), and any kind of information that may be provided directly or indirectly by other (possibly also self-reflecting) systems. Particular challenges arise from the fact that in uncertain environments sensor information may be noisy or missing, communication may be disturbed or delayed, and feedback about the system's own actions cannot or not easily be obtained analyzing sensory information. Humans may, as additional knowledge sources, be part of a self-reflection process. Altogether, self-reflection can be regarded as being founded on self-awareness (knowledge about own knowledge and own behavior, including the past), context-awareness (knowledge about the environment), and mutual awareness (knowledge about other, similar systems).

Clearly, a system of systems (SoS) is again a (distributed) system with particular challenges regarding self-reflection abilities. If such an SoS is not organized in a centralized way, for instance, self-reflection must be realized in a distributed manner by self-organized collaboration techniques. SoS may be heterogeneous and open, i.e., systems may enter or leave the SoS at runtime. With its focus on self-improvement including the definition of new (kinds of) goals, self-reflection in SoS goes beyond adaptation in such systems (cf. the field of collective adaptive systems [24]).

B. A Generic Architecture for Computational Self-Reflection

The architecture shown in Fig. 4 is a blueprint for a technical system with self-reflection abilities. Variants of this architecture are certainly meaningful depending on specific application scenarios, see below. The Organic Computing community has proposed the generic Observer/Controller (O/C) architecture [25] that can be adopted as in Fig. 4 for self-reflecting technical systems. Here, we explicitly add knowledge models (K) to each O/C pair that include appropriate modeling techniques (e.g., machine learning techniques [26]). Basically, this architecture can also be seen as hierarchy of control loops or MAPE-K (monitor, analyze, plan, execute) loops [27].

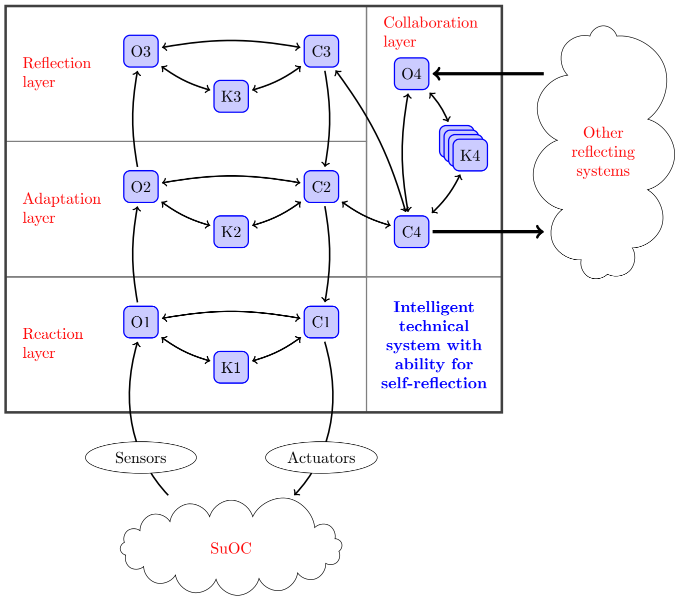

Fig. 4. Reflection architecture for an intelligent technical system.

The *System under Observation and Control* (SuOC) is the environment in which the system is embedded. The SuOC is observed by means of sensors; actuators may influence the processes in the SuOC. As an example, we may consider a robot (reflective system) in a specific environment (SuOC).

The bottom layer (*Reaction Layer*) realizes the standard functionality of the system. The system might have to fulfill real-time constraints. As an example, the O1/C1 pair of components extract appropriate features from sensor signals, categorize these information using classifiers trained from sample data, and trigger appropriate actions for the recognized situation based on knowledge contained in K1.

The *Adaptation Layer* enables the system to autonomously – or semi-autonomously if other systems are involved through the Collaboration Layer – deal with new situations arising at runtime. Here, "new situations" does not mean that the very basic kind of this situation cannot be anticipated; only the point in time when this situation arises cannot be predicted. Moreover, "deal with" means that the system will be enabled to maintain a certain performance degree. For this purpose, the O2/C2 components analyze the new situation relying on more abstract knowledge about the behavior of the Reaction Layer stored in K2 (e.g., experience gained while applying rules at the Reaction Layer), and trigger appropriate actions, e.g., an adaptation of the Reaction Layer's functionality (e.g., in C1), an exchange or modification of K1 (via C1), etc.

The *Reflection Layer* realizes the three building blocks of self-reflecting systems sketched in the above definition in its components O3, K3, and C3. Note that we do not claim that, in general, self-reflecting systems are able to guarantee to always improve their behavior. In special cases, self-reflection may even result in performance degradation. Thus, the Reflection Layer must include techniques to assess the potential gains and risks of adapting the lower layers of the system. We claim that self-reflection must lead to a long term evolution of the system which yields, on average, statistically significant performance improvements. However, any possibly existing hard side conditions (e.g., whenever the system performance is critical and a degradation not acceptable) must be considered. Thus, the Reflection Layer decides whether to let the lower layers proceed with the currently used techniques or whether to trigger their adaptation (e.g., parameters, algorithms, or even structure), for instance.

The *Collaboration Layer*, which can be triggered either by C2 or C3, realizes the communication with other, similar systems and (depending on the application) even humans. It keeps models (K4) about their knowledge (obtained with efficient exploration techniques) and it is able to observe and assess their knowledge and to ask them very specific questions (via O4 and C4).

As already mentioned, variants of this architecture are necessary depending on the application. Examples are

- an architecture with direct connection of the Reaction Layer to the Collaboration Layer (C1 $\leftrightarrow$ C4), e.g., to realize coordinated collective reactions under real-time constraints, or
- an architecture with indirect coupling of systems in a heterogeneous SoS via the SuOC (e.g., if no direct communication is possible).

IV. Application Scenarios

In this section, we discuss three exemplary application scenarios where the usage of self-reflection techniques might be beneficial. These scenarios address different application domains and describe an increasing level of complexity. The first scenario deals with a team of soccer robots that behave as a team. This scenario mainly focuses on heterogeneous hardware sensors and is hence characterized by noisy data. Only small groups of robots (typically 4 to 7) are considered. The second scenario deals with traffic management in urban environments. Compared to a soccer robot, less kinds of sensors are used (i.e., only induction loops) which leads to decreased noise effects. Urban traffic infrastructures consist of larger group sizes in which self-reflection (and reflection about others) can be performed – typically, networks of 10 to 100 intersections are considered. Finally, we take the simulations of natural organisms / systems into consideration. These systems are characterized by less noise and a potentially very large amount of participating entities (several thousand). This section introduces these three scenarios, describes currently available approaches, and explains how self-reflection techniques will lead to a benefit.

A. Scenario 1: Soccer Robots

Scenario: RoboCup is an international joint activity to promote research on autonomous robots. It is an attempt to advance a range of technologies that are needed for mobile autonomous robots that act as a team, including image processing, real-time sensor data processing, sensor fusion, embedded systems, artificial intelligence planning, multi-agent coordination protocols in unreliable environments, and more. RoboCup chose robotic soccer as a central application domain for their research because soccer is well known world-wide, it is a very dynamic and thus challenging game, and the rules of the game are clearly defined and widely accepted. The stated ultimate goal of the RoboCup project is to have by the year 2050 a team of fully autonomous humanoid robots that can beat the human world champion in soccer. Other application domains in RoboCup are search & rescue and service robotics.

In order to act as a team and achieve a common objective, robots need to know about their own current status and task allocation, the status and task allocations of their team members, as well as the status of their environment. All team members contribute their perceptions to a "shared world model" that represents the fused knowledge of the individual team members and is used as a foundation for distributed decision making. Clearly, reasoning and decision making cannot be done in a centralized fashion due to the dynamics of the game and the unreliability of the (wireless) communication environment. Each soccer robot decides by itself what to do, but tries to coordinate its activities with its team mates. From an information model point of view, the shared world model is a set of replicated variables with weak consistency guarantees.

Current approach: A team of soccer robots in RoboCup represents a closed group of collaborating (mostly homogeneous) agents. Their decentralized decision making about strategies and task allocation requires self-reflection, whereby self-reflection happens at the individual robot level, e.g., a robot must recognize its own situation and available action options, as well as at the team level, e.g., the team members must know whether the team as a whole possesses the ball even if individual team members cannot see the ball currently. (The ball may be hidden by other players or it may be out of range for the built-in computer vision.) Obviously, important strategic behavior decisions (such as "attack" or "defend") depend on such a collective awareness at the team-level. Some team decisions require strict consistency (e.g. for a mutually exclusive activity), others tolerate eventual consistency (e.g. the robots' view of the exact position of the ball on the play field may be treated with relaxed consistency requirements because the ball is moving very fast anyway). All of this needs to be considered in an application domain that requires (soft) real-time behavior and where communication is potentially unreliable. Distributed storage, replication, consistency guarantees, and fault tolerance together make maintaining the shared world model a difficult problem. Appropriate systematic support is lacking.

Self-reflection: Data consistency concerns regarding the shared world model in multi-robot systems in highly dynamic environments is only one of many open research questions that are related to self-reflection in such systems. Others are: suitable knowledge representation models, monitoring the effects of collective decisions, on-the-fly learning about success and failure of actions in certain situations, how to re-plan if "the world" changes substantially or too many failed actions happen, and many more. In addition, if we assume open and heterogeneous multi-robot systems, such as in Search & Rescue scenarios, where robots may come and go and where robots of different rescue teams need to collaborate among themselves as well as with humans, a whole new bag of questions opens up. For example, how are the self-reflection models merged if different, possibly heterogeneous, multi-agent teams enter the scene? How do we extend the shared world model on-the-fly? What kind of meta-level domain information is required and how is it maintained? How do we link the robots' view of the world to the know-how and perceptions of human operators? Clearly, there are plenty of challenging open research questions that require further research. Often a rudimentary form of self-reflection is used implicitly in multi-robot systems. However, we need a strong systematic foundation for explicit self-reflection, especially at the team level, in order to facilitate monitoring, awareness, reasoning, decision making, adaptation, learning, collaboration with other teams and human operators, runtime testing and validation, system evolution, and more.

B. Scenario 2: Vehicular Traffic Control

Scenario: Increasing mobility and rising traffic demands cause serious problems in urban road networks – the infrastructure's capacity is limited and current installations face congestion and non-efficient behavior. As a result, research and development investigate solutions for intelligent traffic management systems (ITS). An ITS mainly consists of integrated concepts for traffic-adaptive signalization of traffic lights, coordination of intersections, route recommendations to drivers and incident detection (see, e.g., [28]). The goal is to react as fast as possible and as appropriate as possible to changing traffic conditions, disturbances (i.e., accidents or construction work causing road blockades), and evolving traffic behavior. Typically, the success of solutions is quantified by metrics such as the averaged waiting times (i.e., expressed as Level of Service [29]), the throughput in vehicles per hour, the number of stops when passing a road network, or environmental figures such as pollution.

Current approach: As outlined before, ITS consist of several interconnected components. For traffic signalization, fixed-time control and traffic-actuated control strategies are distinguished. Thereby, either a static behavior is defined or the amount of approaching traffic is estimated based on detector information (mostly perceived by induction loops in the street surface) – which then is considered in the control behavior. The same detectors are also used to derive an overall traffic situation of the network – allowing for further higher-level actions such as route recommendation to drivers. Typically, all current installations are characterized by non-reflective solutions (see, e.g., [30]). Detector data is smoothed and analyzed, but reasoning in the sense of correcting noisy detector data or recognizing recurring patterns has not been applied yet. For instance, such information would be necessary to automatically detect incidents. In those installations where traffic condition estimation is processed, mostly high-level stream information is taken into account – this is not suitable for inner-city traffic management. Additionally, switching between goals as reaction to the particular traffic conditions is not considered. For instance, high traffic conditions demand for an increase of the throughput, while low traffic conditions should be handled by decreasing the averaged waiting times (these goals are contrary, see [31]).

Self-reflection: A self-reflective approach can go far beyond the current approaches. Instead of just smoothing and utilizing sensor data to determine the current system status, a consistent and more appropriate description of the current situation can be generated. The potential benefit is to derive and adjust models of the road segments and their traffic conditions – for instance, including models to estimate the existence and the status of possible parking spaces.

Therefore, intersection controllers need further abilities. First, they have to monitor and to assess a) their own behavior and status, b) behavior and status of neighboring intersection controllers, and c) the environment (e.g., roads in-between, detector capabilities, etc.). For all these influencing entities, models have to be derived that are consistent with the current detector readings. Based on this, intersection controllers can collaboratively reason about the current status – for instance, detect parking spaces between them where vehicles are buffered for a certain period.

Second, these models have to incorporate the experiences with utilizing them and consequently the certainty about their correctness. Considering the previous parking space example, deviations between leaving and arriving vehicles between intersections connected with a road can be also caused by incidents decreasing the road's capacity. Finally, the intersection controllers have to decide about their current and long-term goals. Considering the varying aforementioned goals for traffic control, intersection controllers have to decide about the best strategy in the presence of conflicting goals. In the context of intelligent traffic management, self-reflection at the level of individual intersection controllers, intermediary regions, and at the system level promise significant improvements regarding situation assessment and runtime decisions (i.e., route guidance).

C. Scenario 3: Simulation of Natural Processes

Scenario: The incessant and multi-facetted empiric investigation of natural processes has unearthed patterns, hypotheses and laws at various degrees of abstraction. Their integration – as disparate their representation, as deviating their scales – has culminated in multi-scale approaches that link otherwise independent models such as black-boxes, intertwining results of some with the parameter sets of others (for a review of multi-scale modeling techniques see, for instance, [32]). Bottom-up models pose a powerful alternative as they yield system properties observed at higher levels of abstraction as emergent effects of underlying interaction processes [33]. The need for comprehensive model data at an evenly fine-grained level of abstraction, which is often realized by means of agent-based representations, is accompanied by other inherent challenges that arise from bottom-up design. For instance, the models' emergent properties need to be (top-down) consistent with the predictions of the respective higher-level stand-alone models [34]. Broad low-level model integration also implies great computational costs due to open, i.e. not fully predetermined, behaviors of the consequently large number of model components.

Current Approach: Model adaptation is an emerging trend for scaling agent-based models, as it often possible to reduce the (initially inherently great) model complexity based on observation of the resultant simulation processes. In one way or another, agents are reduced in their individual complexity or merged to reduce the overall complexity of the model. In [35], an approach is described that clusters and merges agents based on principle component analysis of their attributes. Instead of statically merging agents, it may be beneficial to manage their subsumption dynamically based on current model querries [36]. In [37], one of the authors presented a concept where observer agents would be immersed into a simulation and proactively seek interaction patterns to simplify the system model during runtime. Due to the distributed nature of the model, the necessary changes to the model could be effected locally by the observer agents themselves. Recognized patterns might be transient and the learned simplifications and the implemented abstractions might, therefore, become invalid after some event or after a sequence of events – certainly after some period of time. Therefore, confidence [38] is built up over time and periods of validation of previously made abstractions grow accordingly. Validation is performed by releasing agents subsumed by an abstracting meta-agent back into the simulation and testing whether their behavior is in line with the predictions. If prediction and actual lower level behavior deviate, the abstraction is revoked and the lower-level agents become part of the simulation model again. On the other hand, if the abstraction holds, the meta-agent itself becomes subjected to observation and potentially to further abstraction by an observer agent. Thus, the system model is represented as a set of dynamic hierarchies of meta-agents. Aspects of this approach to self-organized model adaptation have been implemented in the context of gene signaling pathways and blood coagulation processes, implementing different means for pattern detection (from co-variance measures to process frequencies) and agent subsumption (approximation with artificial neural networks or rule-based meta-behaviors.

Self-reflection: Self-organized model adaptation necessitates the aforementioned capacities of self-monitoring, self-assessment, and self-improvement (Section I). At this point, there are implementations that address these requirements in different ways, but a clear implementation strategy, a clear choice for specific methods has not materialized, yet. The idea of a distributed model, as provided by agent-based approaches, promotes locally performed observation and locally implemented model abstractions – as a consequence, numerous "selves" reflect upon the system model and modify it in accordance with their observations. Similarly to collaborative self-organizing systems, challenges of coordination and opportunities for fruitful imitation and synergy arise. *The* self-reflection of the whole system model, however, could emerge from observation of all the deployed agents, observers, and meta-agents and from analysis of their (dynamic) interplay. In the context of large-scale agent-based simulations, self-reflection at the level of individual model agents, intermediary model clusters, and at the system level promise significant improvements regarding simulation runtime and accuracy.

V. Building Blocks for Technical Self-Reflection

Self-reflection is an open field of research. Thereby, research can make use of results and methods from several existing areas. This section identifies possible contributions from related fields and describes possible solution perspectives to develop self-reflective systems. As already considered in the motivating application scenarios, self-reflective systems are specific instances from the group of self-organizing systems. As a basic for the design of self-reflective systems, architectural concepts from the Autonomic and Organic Computing domains can be adapted. Within such an architecture, self-reflective solutions will make heavy use of models – either generated at design-time or at runtime. Hence, initiatives like Models@Runtime will play a major role. Since system behavior and models are based on incomplete and noisy sensor data, concepts to deal with uncertainty, pattern recognition, emergence, and anomaly detection are required – these are covered by, e.g., data mining or probabilistic theory. The potentially most important capability of self-reflective systems will be to define new goals and new classes of goals in order to increase its own performance. In order to find novel ways for goal strategies, solutions might again make use of the aforementioned techniques – since this is a new research direction, further re-usable methods have to be identified. Finally, the result of self-reflection in cooperative groups of systems will result in collections of self-adaptive and self-reflective systems – which is closely related to collective adaptive systems and the corresponding theory.

A. *Organic Computing*

Organic Computing (OC) [39] and similar scientific initiatives such as Autonomic Computing [27] postulate that integrative design of complex, open systems demands for a paradigm shift in engineering. Instead of designers anticipating all possible system configurations at design-time, a system needs to be flexible and be empowered by great degrees of freedom. Only then, the system is enabled to manage itself and to adjust itself to changing conditions at runtime – meaning that decisions that are typically made at design-time are now addressed by a system's runtime components. Since natural systems are typically characterized by highly robust and adaptive solutions, OC investigates means to transfer nature-inspired processes to technical systems. The goal is to develop life-like properties for technical systems. In the context of self-reflective systems, OC can provide several mechanisms that can find their way into future solutions. Especially the architectural concept such as the generic *Observer/Controller pattern* [25] or the MAPE-K cycle [27] serve as platform for further research (see Section III-B). Additionally, OC investigated mechanisms to detect emergent effect in self-organized systems. Methods for the quantification of emergence such as in [40] can serve as a basis for novelty, anomaly or obsoleteness detection. Furthermore, modifications and extensions of these concepts might be useful to model aspects such as concept drift.

B. *Models@Runtime*

Self-reflection relies on the system's capability to automatically build models of itself, other systems, and the environment. The research field of Models@Runtime started with the motivation that model-driven engineering processes generate varying models at design-time which are not further used at the system's runtime. Hence, possibilities to extend the applicability of models and abstractions during operation of the system to be developed are investigated. The motivation of the conducted research is to find ways for providing effective technologies for mastering and managing the complexity of evolving software behavior. Thereby, not just the design-time – but also the runtime aspects can be taken into account making solutions more appropriate [41]. Since Models@Runtime is a very heterogeneous initiative, several concepts from different fields (such as model-driven development processes, requirements engineering, or validation and verification) can serve as input to investigate self-reflective systems. Especially work on the generation of behavioral models for dynamic adaptive systems [42] provides a good starting point.

C. *Machine Learning*

Machine Learning (ML) obviously plays a key role in self-reflection. Of particular interest are techniques that allow for a high degree of autonomy in learning. Assuming that (fully) supervised learning is not possible, we may rely on techniques for reinforcement learning or semi-supervised learning, for instance. In the field of OC, ML techniques are used quite frequently as building blocks for many applications. To keep user feedback at a minimum, learning is accomplished by different approaches as, for instance, exchanging information (e.g., decision rules) between components in a distributive system [43], [44], using active learning techniques to effectively and efficiently integrate human domain experts [45], simulating the environment [28], or imitating the behavior of other systems [46].

Introspection and intercession challenge a system to discover patterns, to draw the right conclusions and to effect appropriate changes. The successful pursuit of these steps greatly depends on the computational representations and the model building approaches that mediate between them. Ideally, a single representation would lend itself to all three steps and, in addition, be flexible in terms of data types and value ranges, be discriminative, human-readable, offer precise extrapolation for complex data sets, and, most importantly, support the generation, combination and incremental variation of model hypotheses. One of the authors has proposed the utilization of polynomials for approximation of time series—not only are there efficient fitting algorithms but they also allow for aggregate, hierarchical composition [47]. Combinations of generative and discriminative models, for instance classification based on kernel functions [48] or tracking based on latent appearances [49] clearly show that an integrative approach is feasible. Most frequently, working with learned meta-data puts forth two-tier approaches to representation, where the meta-data management and meta-data contents are only loosely coupled, if at all (see for instance [35] or [50]). As a self-reflective technical system needs to adjust itself to specific circumstances and conduct specialized actions just as much as it is required to learn generalizations from experience that may equally improve its performance and adaptivity, the search for generic, efficient and expressive representations is another important aspect of self-reflection in technical systems.

D. Probability and Possibility Theory

Uncertainty is a general term that refers to missing, noisy, or possibly wrong information, for instance. We are uncertain concerning the exact values of observations (sensor measurements) from the SuOC, concerning the exact parametrization of our knowledge models at various levels, concerning the exact time stamps of events due to communication problems (e.g., delays) etc. Methods from possibility theory or probability theory (e.g., type-II fuzzy systems, Dempster-Shafer theory of beliefs [51], [52], or second-order probability distributions [53]) can be used to model various kinds of uncertainty at various levels. A numerical quantification of uncertainty will support decision processes, e.g., if potential gain or risk of possible decisions can be estimated.

E. Data Mining

The concept of "knowledge about knowledge" is also known in the field of *data mining* [54]. Data mining can be seen as a multi-step process as shown in the data mining pyramid (see Fig. 5) which has been introduced by Embrechts et al. in [55]. The idea of this pyramid can briefly be summarized as follows: Raw *data* are pre-processed to condense application-specific *information* in attributes or features. Then, *knowledge* is extracted, e.g., by building classification or regression models. By analyzing this knowledge off-line and later on by using it in a given application (on-line) it is possible to come to a deeper *understanding* of its working principles and to gain some *experience* in using it, respectively. Both will support the efficient and effective application of the knowledge. Finally, this kind of meta-knowledge will eventually help to solve similar kinds of application problems. That is, the final step of *wisdom* is reached by transferring the knowledge to other application domains.

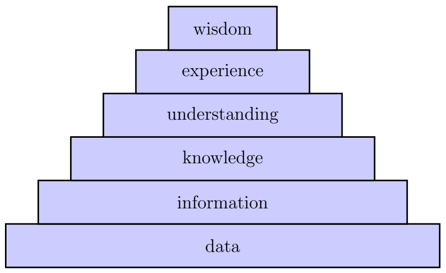

Fig. 5. The data mining pyramid (adopted from [55]).

F. Dependency Modeling

Self-reflective systems can work in cooperation with other systems. Thereby, dependencies between these systems can exist. Consider the soccer robots as introduced in Section IV-A: If robot A is covering the role of a goalkeeper, this role is not available for other team mates. Hence, dependency modeling is needed for self-reflective systems as it supports the environment-awareness. These dependencies can be explicit in terms of acting on simple parameter setting (such as synchronized phases in traffic control to establish Progressive Signal Systems), on group structures (e.g., cooperation with entity X prohibits cooperation with entity Y), or on organization schemes (i.e., in the robot scenario). Furthermore, implicit and hidden dependencies can exist that are much harder to detect. For instance, the quality of the decision to route vehicular traffic via a certain link depends highly on decisions of other nodes in the network to route their traffic to a certain degree via different links, otherwise traffic jams can appear (the locality of this impact is not restricted to the direct neighborhood). For explicit dependencies, the knowledge models of preliminary work has been mostly adequate, since self-learning systems can find such relations based on, e.g., reinforcement learning techniques. For implicit dependencies, methods to model dependencies between randomized variables are needed. For instance, risk measures [56] from Operational Research or copula models [57].

G. Collective Adaptive Systems

A Collective Adaptive System is a collection of heterogeneous autonomous entities that have autonomous goals and behaviors, but cooperate with one another and adapt collectively in order to accomplish their own tasks or a common task and reach their individual goals or a common goal in an efficient and effective way [24]. It can be seen as the next step and logical continuation from context-awareness and self-adaptation of a single application or system. Collective adaptability based on information and communication technology (ICT) is targeted at large scale distributed systems that are composed of large numbers of interacting and heterogeneous hardware / software components [58]. Moreover, Collective Adaptive Systems typically operate with a high degree of autonomy, under varying and often unpredictable conditions, and in open dynamic environments. Often, these systems become more and more embedded in the fabric of organizations and society at large. Thus, they are used by a large number and a wide variety of users, which makes them true socio-technical systems [59]. Collective adaptation is often viewed as a prerequisite for Collective Intelligence [60].

In order to reason about and execute collective adaptation activities participants need a shared knowledge base, i.e., collective action requires self-reflection. Based on the shared knowledge and a given objective function, individual actors will decide about their contributions to collective activities. Objective functions that operate on the self-reflective model may be specified as, e.g., utility functions, constraints, or rule-based. Scalability of the underlying self-reflection mechanisms is a must, but often difficult to realize and to prove. Monitoring and measuring the degree of success of collective adaptations is a difficult problem. Likewise, the collective interaction and adaptation will likely lead to emergent system properties and behavioral patterns that need to be understood and influenced. If we focus on the socio-technical aspects it becomes obvious that self-reflection and collective adaptation can involve the collecting and processing of large amounts of sensible personal user data. These are just a few open questions that need to be addressed by research.

VI. Research Agenda

The previous section motivated and described several different building blocks as basis for research on self-reflective systems. Based on this, we define a research agenda towards fully self-referential systems that includes open questions to be addressed. This research agenda is aligned along the architecture as illustrated in Figure 4. The major research issues are concerned with Reflection and Collaboration Layer, while the work on Reaction and Adaptation Layer will benefit from preliminary work. The remainder of this section discusses the topics to be investigated by following the architectural layers in increasing order.

Reaction Layer: The Reaction Layer realizes the reactive modification of the system's behavior. Thereby, research will mainly focus on building appropriate knowledge models. This is accompanied by work on quantifying the degree of certainty about the particular models representing the own, the external and the environmental behaviors. Additionally, concepts to choose the desired actions (e.g., parameter configurations, selection of strategies, etc.) have to take this information into account. Furthermore, deviations between model and reality have to be detected as fast and as appropriate as possible.

Adaptation Layer: Since this layer enables the system to autonomously deal with new situations arising at runtime, mechanisms to maintain a certain performance degree have to be available. Besides the existing work (especially in the context of OC), novel methods to quantify the uncertainty about perceived information are needed. For this purpose, models (the K2 component in Figure 4) have to be adapted online. The most important questions arising here are: How to detect relations in behavior and performance at runtime and without prior knowledge.

Reflection Layer: This layer incorporates the basic self-reflection capability by considering the three general aspects of reflection. Thus, research dealing with the Reflection Layer has to be able to assess the potential gains and risks of available adaptation strategies for the lower layers of the system. Additionally, we need mechanisms to (cooperatively with others) build models of the possible behaviors and quantify their benefit and the corresponding risk. This has is accompanied by the need to detect dependencies between individual systems – resulting in an distributed optimization process with several participating individuals. Thereby, cooperative runtime modeling plays an important role. Furthermore, the major issues of self-reflection are located at this layer: How can systems themselves define new goals and new classes of goals in order to increase their own performance? This is accompanied by the need to identify appropriate techniques that can be utilized to find novel ways for goal strategies – resulting in the question of how to modify them and what is additionally needed to fulfill the requirements in developing novel goals. Such a self-managed goal adaptation and generation process has to be guided – and obviously controlled at a more abstract level. Hence, questions arise how to guarantee a certain behavior even in the presence of changing goals – and how to control the resulting behavior in case checking the accordance with a pre-defined goal is not an option, any more.

Collaboration Layer: Here, the goal is to develop new basic technologies to realize collective self-reflection. A collection of autonomous self-organized systems does not necessarily require that each system belongs to the same authority. Hence, we cannot assume that knowledge can be easily transferred from one entity to another. In this context, concepts for abstract knowledge representations, demand-oriented exchange of these, and a common "language" for self-reflective systems are needed.

Acknowledgment

The authors would like to thank the German Research Foundation (DFG) for the support of the project CYPHOC. The work of Bernhard Sick and Christian Gruhl is partly funded under reference (SI 674/9-1), while the work of Sven Tomforde and Jörg Hähner refers to reference (HA 5480/3-1).

References

[1] A. Scholtz, "Gnōthi sauton – "know thyself"," online: http://harvey.binghamton.edu/~grk101/gnothi_sauton.pdf, Binghamton University, Department of Classical and Near Eastern Studies, Binghamton, NY, 2006, (last access: 06/04/2014).

[2] F.-P. Hager, "Selbsterkenntnis," in *Historisches Wörterbuch der Philosophie*, J. Ritter, K. Gründer, and G. Gabriel, Eds. Basel, Switzerland: Schwabe, 1971 – 2007, vol. 9, pp. 406 – 413.

[3] B. Gertler, "Self-knowledge," in *The Stanford Encyclopedia of Philosophy*, spring 2011 ed., 2011, (online: http://plato.stanford.edu/archives/spr2011/entries/self-knowledge/, last access: 06/04/2014).

[4] R. W. Henke, "Selbsterkenntnis," in *Handwörterbuch Philosophie*, W. D. Rehfus, Ed. Stuttgart, Germany: UTB, 2003.

[5] W. D. Rehfus, "Reflexion," in *Handwörterbuch Philosophie*, W. D. Rehfus, Ed. Stuttgart, Germany: UTB, 2003.

[6] L. Zahn, "Reflexion," in *Historisches Wörterbuch der Philosophie*, J. Ritter, K. Gründer, and G. Gabriel, Eds. Basel, Switzerland: Schwabe, 1971 – 2007, vol. 8, pp. 396 – 405.

[7] Wikipedia, "Human self-reflection — wikipedia, the free encyclopedia," online: http://en.wikipedia.org/wiki/Human_self-reflection, 2014, (last access: 06/04/2014).

[8] L. Wood, "Reflection," in *The Dictionary of Philosophy*, D. D. Runes, Ed. New York, NY: Philosophical Library, 1942.

[9] M. Koch, "Introspektion," in *Historisches Wörterbuch der Philosophie*, J. Ritter, K. Gründer, and G. Gabriel, Eds. Basel, Switzerland: Schwabe, 1971 – 2007, vol. 4, pp. 522 – 524.

[10] E. Schwitzgebel, "Introspection," in *The Stanford Encyclopedia of Philosophy*, summer 2014 ed., 2014, (online: http://plato.stanford.edu/archives/sum2014/entries/introspection/, last access: 06/04/2014).

[11] Wikipedia, "Introspection — wikipedia, the free encyclopedia," online: http://en.wikipedia.org/wiki/Introspection, 2014, (last access: 06/04/2014).

[12] D. A. Schön, *The Reflective Practitioner, How Professionals Think In Action*. New York, NY: Basic Books, 1983.

[13] G. Gibbs, "Learning by doing, a guide to teaching and learning methods," online: http://www2.glos.ac.uk/gdn/gibbs/index.htm, 1988, (reproduced by The Geography Discipline Network, last access: 06/20/2014).

[14] D. Kolb, *Experiential Learning: experience as the source of learning and development*. Englewood Cliffs, NJ: Prentice Hall, 1984.

[15] C. Argyris and D. Schön, *Organizational Learning: A Theory of Action Perspective*. Reading, MA: Addison Wesley, 1978.

[16] J. Cowan, *On Becoming an Innovative University Teacher: Reflection in Action*, 2nd ed. Berkshire, U.K.: Society for Research into Higher Education & Open University Press, 2006.

[17] S. Brookfield, "Critically reflective practice," *Journal of Continuing Education in the Health Professions*, vol. 18, pp. 197 – 205, 1998.

[18] B. C. Smith, "Procedural reflection in programming languages," Ph.D. dissertation, MIT, 1982.

[19] P. Maes, "Concepts and experiments in computational reflection," in *ACM Sigplan Notices*, vol. 22, no. 12, 1987, pp. 147 – 155.

[20] É. Tanter, "From metaobject protocols to versatile kernels for aspect-oriented programming," Ph.D. dissertation, University of Nantes and University of Chile, Nov. 2004.

[21] M. Rehák, M. Pěchouček, and M. Rollo, "An abstract architecture for computational reflection in multi-agent systems," in *Intelligent Agent Technology*, no. PR2416 IEEE, Los Alamitos, 2005.

[22] K. Bellman and C. Landauer, "Reflection Processes Help Integrate Simultaneous Self-Optimization Processes," in *ARCS 2014 – Feb. 25-28, 2014, Lübeck, Germany - Workshop Proc.*, 2014, pp. 1 – 5.

[23] C. Landauer, "Infrastructure for studying infrastructure," in *Presented as part of the 2013 Workshop on Embedded Self-Organizing Systems (last access: 07/14/2014)*. Berkeley, CA: USENIX, 2013. [Online]. Available: (https://www.usenix.org/conference/esos13/workshop-program/presentation/Landauer)

[24] S. Kernbach, T. Schmickl, and J. Timmis, "Collective adaptive systems: Challenges beyond evolvability," *ACM Computing Research Repository (CoRR)*, 2011, last access: 07/14/2014. [Online]. Available: ftp://ftp.cordis.europa.eu/pub/fp7/ict/docs/fet-proactive/shapefetip-cas09_en.pdf

[25] S. Tomforde, H. Prothmann, J. Branke, J. Hähner, M. Mnif, C. Müller-Schloer, U. Richter, and H. Schmeck, "Observation and Control of Organic Systems," in *Organic Computing – A Paradigm Shift for Complex Systems*. Basel, CH: Birkhäuser Verlag, 2011, pp. 325 – 338.

[26] C. Müller-Schloer, H. Schmeck, and T. Ungerer, Eds., *Organic Computing – A Paradigm Shift for Complex Systems*. Basel, CH: Birkhäuser Verlag, 2011.

[27] J. O. Kephart and D. M. Chess, "The Vision of Autonomic Computing," *IEEE Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[28] H. Prothmann, S. Tomforde, J. Branke, J. Hähner, C. Müller-Schloer, and H. Schmeck, "Organic Traffic Control," in *Organic Computing – A Paradigm Shift for Complex Systems*. Basel, CH: Birkhäuser Verlag, 2011, pp. 431 – 446.

[29] National Electrical Manufacturers Association, "NEMA Standards Publication TS 2-2003 v02.06 – Traffic Controller Assemblies with NTCIP Requirements," Rosslyn, Virginia, USA, 2003.

[30] A. W. Sadek, "Artificial Intelligence Applications in Transportation," *Transport Research CIRCULAR*, vol. EC-113, 2007.

[31] S. Tomforde, H. Prothmann, J. Branke, J. Hähner, C. Müller-Schloer, and H. Schmeck, "Possibilities and limitations of decentralised traffic control systems," in *2010 IEEE World Congress on Computational Intelligence (IEEE WCCI 2010), July 18-23, 2010. Barcelona, Spain*. IEEE, 2010, pp. 3298–3306.

[32] M. Horstemeyer, "Multiscale modeling: A review," *Practical Aspects of Computational Chemistry*, pp. 87–135, 2010.

[33] C. Jacob, S. von Mammen *et al.*, *LINDSAY Virtual Human: Multi-scale, Agent-based, and Interactive*, ser. Advances in Intelligent Modelling and Simulation: Artificial Intelligence-based Models and Techniques in Scalable Computing. Berlin / Heidelberg, DE: Springer Verlag, 2012, vol. 422, pp. 327–349.

[34] D. Noble, *The music of life*. Oxford University Press Oxford, 2006.

[35] A. R. Stage, N. L. Crookston, and R. A. Monserud, "An aggregation algorithm for increasing the efficiency of population models," *Ecological modelling*, vol. 68, no. 3, pp. 257–271, 1993.

[36] S. Wendel and C. Dibble, "Dynamic agent compression." *Journal of Artificial Societies & Social Simulation*, vol. 10, no. 2, 2007, last access 14/07/2014. [Online]. Available: http://jasss.soc.surrey.ac.uk/10/2/9.html

[37] S. von Mammen, J.-P. Steghöfer, J. Denzinger, and C. Jacob, "Self-organized middle-out abstraction," in *Self-Organizing Systems*, ser. LNCS, vol. 6557. Karlsruhe, DE: Springer, 2011, pp. 26–31.

[38] J. Kiefer, "Conditional confidence statements and confidence estimators," *Journal of the American Statistical Association*, vol. 72, no. 360, pp. 789–808, 1977.

[39] C. Müller-Schloer, "Organic Computing: On the Feasibility of Controlled Emergence," in *CODES+ISSS'04 Proceedings of the 2nd International Conference on Hardware/software codesign and system synthesis*. New York, USA: ACM, 2004, pp. 2–5.

[40] D. Fisch, M. Jänicke, B. Sick, and C. Müller-Schloer, "Quantitative Emergence – A Refined Approach Based on Divergence Measures," in *Proc. of 2010 4th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO), Sept. 27 2010-Oct. 1 2010, Budapest, HU*, 2010, pp. 94–103.

[41] U. Assmann, N. Bencomo, B. Cheng, and R. France, "Models@run.time," *Dagstuhl Reports*, vol. 1, no. 11, pp. 91 – 123, 2011.

[42] H. J. Goldsby and B. Cheng, "Automatically Generating Behavioural Models of Adaptive Systems to Address Uncertainty," in *Proc. of Model Driven Engineering Languages and Systems. 11th International Conference, MoDELS 2008, Toulouse, France, September 28 - October 3*, ser. LNCS 5301. Springer, 2008, pp. 568 – 583.

[43] D. Fisch, M. Jänicke, E. Kalkowski, and B. Sick, "Learning from others: Exchange of classification rules in intelligent distributed systems," *Artificial Intelligence*, vol. 187, pp. 90–114, 2012.

[44] ——, "Techniques for knowledge acquisition in dynamically changing environments," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 7, no. 1, pp. 1–25, 2012.

[45] T. Reitmaier and B. Sick, "Let us know your decision: Pool-based active training of a generative classifier with the selection strategy 4DS," *Information Sciences*, vol. 230, pp. 106–131, 2013.

[46] A. Jungmann, B. Kleinjohann, and W. Richert, "Increasing learning speed by imitation in multi-robot societies," in *Organic Computing – A Paradigm Shift for Complex Systems*. Basel, CH: Birkhäuser, 2011, pp. 295–307.

[47] E. Fuchs, T. Gruber, J. Nitschke, and B. Sick, "Online segmentation of time series based on polynomial least-squares approximations," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PP, no. 99, pp. 1 –1, 2010.

[48] T. Jaakkola, D. Haussler *et al.*, "Exploiting generative models in discriminative classifiers," *Advances in Neural Information Processing Systems 12 – NIPS 1999, November 29 - December 4, Denver, USA*, pp. 487–493, 1999.

[49] R.-S. Lin, D. A. Ross, J. Lim, and M.-H. Yang, "Adaptive discriminative generative model and its applications," in *Advances in Neural Information Processing Systems 17 – Neural Information Processing Systems 2004, December 13-18, Vancouver, Canada*, 2004, pp. 801–808.

[50] A. S. Shirazi, S. von Mammen, and C. Jacob, "Abstraction of agent interaction processes: Towards large-scale multi-agent models," *Simulation*, vol. 89, no. 4, pp. 524–538, 2013.

[51] G. Shafer, *A Mathematical Theory of Evidence*. Princeton: Princeton University Press, 1996.

[52] A. Dempster, "Upper and lower probabilities induced by a multivalued mapping," *The Annals of Mathematical Statistics*, vol. 38, no. 2, pp. 325 – 339, 1967.

[53] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY: Springer, 2006.

[54] D. Fisch, E. Kalkowski, and B. Sick, "In your interest: Objective interestingness measures for a generative classifier," in *Proceedings of the 3rd International Conference on Agents and Artificial Intelligence (ICAART 2011)*, Rome, Italy, 2011, pp. 414 – 423.

[55] M. J. Embrechts, B. Szymanski, and K. Sternickel, "Introduction to scientific data mining: Direct kernel methods and applications," in *Computationally Intelligent Hybrid Systems*, S. J. Ovaska, Ed. Piscataway, NJ: IEEE Press, 2005, ch. 10, pp. 317–362.

[56] A. J. McNeil, R. Frey, and P. Embrechts, *Quantitative risk management: concepts, techniques and tools*. Princeton University Press, 2005.

[57] R. Nelsen, *An Introduction to Copulas (Lecture Notes in Statistics)*. New York, NY, USA: Springer, 1998.

[58] L. Northrop, P. Feiler *et al.*, "Ultra-Large-Scale Systems: The Software Challenge of the Future." Software Engineering Institute, Carnegie-Mellon, Tech. Rep., 2006, last access: 07/14/2014. [Online]. Available: http://www.sei.cmu.edu/uls/

[59] D. Robertson, S. Anderson, I. Carreras, and D. Miorandi, "Social-ist: D2.1 white paper on research challenges in social collective intelligence," http://smart.inf.ed.ac.uk/social-ist-d2-1-white-paper-on-research-challenges-in-social-collective-intelligence/, last access: 07/14/2014, Tech. Rep., 2012.

[60] F. Sestini, "Collective Awareness Platforms: Engines for Sustainability and Ethics," *IEEE Technology and Society Magazine*, pp. 54 – 62, 2012.