On the Definitions of Self-Managing and Self-Organizing Systems

Gero Mühl[1], Matthias Werner[1], Michael A. Jaeger[1*],
Klaus Herrmann[2], and Helge Parzyjegla[1**]

[1] Berlin University of Technology
Einsteinufer 17, Sekretariat EN6, 10587 Berlin, Germany
[2] University of Stuttgart
Universitätsstr. 38, IPVS, 70569 Stutgart, Germany
{g_muehl, m_werner, michael.jaeger, klaus.herrmann, parzyjegla}@acm.org

Abstract. The management costs of software systems are becoming the dominating cost factor in running IT infrastructures. The main driving force behind this development is the ever-increasing system complexity which is becoming the limiting factor for further development. The fact that human administrators get more and more overstrained by management tasks has led to the idea of systems that manage themselves, i.e., self-managing systems. Another concept that is closely related to self-management is self-organization. Self-organizing software systems often build on bio- and nature-inspired approaches. However, most publications on self-managing or self-organizing systems miss a clear definition of these terms. Even worse, although self-management and self-organization aim at similar goals, their relation still has not been defined properly.
In this paper, we approach these problems by introducing a classification of systems that models self-organizing systems as a subclass of self-managing systems. The classification builds upon a definition of adaptive systems and introduces self-manageable, self-managing, and self-organizing systems. Our proposal serves as a starting point for further discussions, eventually leading to a better understanding of the terms self-organization and self-management and their interrelationship.

1 Introduction

Software systems are getting increasingly complex, and their complexity is about to become the limiting factor for further developments. This situation is also termed "complexity crisis" [1] in reference to the software crisis that set in about 40 years ago. In the recent past, self-managing and self-organizing systems have been recognized as being the silver bullet for overcoming this crisis. Thus, it is not surprising that the terms *self-management* and *self-organization* are currently found in the titles of many computer science publications. However, most publications only give informal definitions of these terms which provide little

* Funded by Deutsche Telekom Stiftung
** Funded by the Deutsche Forschungsgemeinschaft (DFG) under grant SPP 1183

more than a very basic intuitive understanding. As a direct consequence, it is hard to unambiguously decide whether or not a given system is self-organizing or self-managing. Moreover, there is no scientific work that describes the relation between self-managing and self-organizing systems.

In previous work [2,3], we have established a definition of *self-organizing software systems*. This definition builds upon the definition of adaptive systems as introduced by Zadeh [4]. We have introduced a methodology which allows to clearly and verifiably state whether and why a given system is self-organizing or not. In this paper, we provide a definition of *self-managing systems* that also relies on Zadeh's concept of adaptivity. In addition, we discuss how adaptive, self-managing, and self-organizing systems relate to each other. Our proposal serves as a starting point for further discussions, eventually leading to a better understanding of the terms self-organization and self-management and their interrelationship.

The remainder of this paper is structured as follows: Section 2 introduces our system model. Adaptive systems are discussed in Sect. 3. Building upon the definition of adaptive systems, we introduce self-manageable and self-managing systems in Sect. 4. Section 5 presents the definition of self-organizing systems from [3] in the context of the presented approach. Section 6 discusses the relation between the different system sets and presents a containment hierarchy of these sets. In Sect. 7, our classification is applied to a number of example systems. Finally, Sect. 8 concludes the paper.

2 System Model

In this section, we define our system model which is based on a behavioral system model by Willems [5].

Definition 1 (System). *A* system $\mathfrak{S}$ *is a tuple* $\mathfrak{S} = (\mathcal{I}, \mathcal{O}, B)$ *with input interface* $\mathcal{I}$*, output interface* $\mathcal{O}$*, and behavior* B*.*

The *input interface* $\mathcal{I}$ defines the inputs of the system, while the *output interface* $\mathcal{O}$ defines the outputs of the system. An *input function* $i(t) : \mathbb{T} \to dom(\mathcal{I})$ is a time-dependent vector-valued function, where $dom(\mathcal{I})$ is the range of values of $\mathcal{I}$. Usually, $\mathbb{T}$ is assumed be equal to $\mathbb{R}_0^+$ (in case of a continuous time) or $\mathbb{Z}_0^+$ (in case of a discrete time). So we do for the rest of this paper. Let $\mathbb{I}$ be the set of all input functions. In the following, we assume without loss of generality that a system has no useless inputs. A *useless input* is an input which does not influence the output. An *output function* $o(t) : \mathbb{T} \to dom(\mathcal{O})$ is also a time-dependent vector-valued function, where $dom(\mathcal{O})$ is the range of values of $\mathcal{O}$. Let $\mathbb{O}$ be the set of all output functions.

The *behavior* of a system is a total relation $B : \mathbb{I} \to \mathbb{O}$ which assigns to each $i \in \mathbb{I}$ at least one $o \in \mathbb{O}$. Thus, if a system is fed with a specific input function i, it will output one of the output functions o for which $(i, o) \in B$. We also say that (i, o) is a *possible behavior* of a system if $(i, o) \in B$. A system is *non-deterministic* iff there is at least one i for which $(i, o) \in B$ for more than one o. Otherwise, the system is *deterministic*. In the latter case, B degrades to a total function, and we can write $o = B(i)$ instead of $(i, o) \in B$. For the sake of simplicity, we restrict the discussion to deterministic systems in the following. However, a similar argumentation is also possible for non-deterministic systems.

The above definition of a system's behavior does not exclude that the output of a system anticipates the input of the system. For example, a behavior may assign to an input function $i(t) = f(t)$ an output function $o(t) = f(t + 10)$, i.e., $o(t) = i(t + 10)$. However, in our scenario this is an important constraint that we formalize in the following definition. For this purpose, we denote the series of values of a time-dependent function f for all $t' \leq t$ as $f_{|t}$ and define that $f_{|t} = f'_{|t} \Leftrightarrow \forall t' \leq t : f(t') = f'(t')$.

Definition 2 (Nonanticipating Behavior). *A behavior B is* non-anticipating *iff*

$$\forall(t \in \mathbb{T}, i \in \mathbb{I}, i' \in \mathbb{I}) : i_{|t} = i'_{|t} \Rightarrow B(i)_{|t} = B(i')_{|t}.$$

The above definition states that if two input functions are identical up to some t, then their corresponding output functions are also identical up to this t. Thus, the output for time t can only depend on the input for times $t' \leq t$. In our definition of self-manageable systems, we will use the concept of a nonanticipating behavior (or function) to restrict the set of possible control functions.

3 Adaptive Systems

Intuitively, we expect an adaptive system to perform acceptably well for various input functions. But how can the fact that a system "performs acceptably well" be stated formally? Following the definition of adaptivity given by Zadeh [4], we formally test whether a system behaves acceptably well for an input function i by using a performance function p (which evaluates the "performance" of the system when it is subjected to i) and an acceptance criterion $\mathcal{W}$ (which checks whether the exhibited "performance" is acceptable). A *performance function* $p : \mathbb{I} \times \mathbb{O} \rightarrow (\mathbb{T} \rightarrow \mathbb{R}^n)$ takes an input function and an output function as parameters and returns a time-dependent vector-valued function with real-valued components. An *acceptance criterion* $\mathcal{W}$ is a set of time-dependent functions f with $f : \mathbb{T} \rightarrow \mathbb{R}^n$. Now, we can define what it means for a system to be adaptive:

Definition 3 (Adaptive System). *A system $\mathfrak{S} = (\mathcal{I}, \mathcal{O}, B)$ is* adaptive *wrt. a set $I \subseteq \mathbb{I}$, a performance function p, and an acceptance criterion $\mathcal{W}$ iff $\forall(i \in I) : o = B(i) \Rightarrow p(i, o) \in \mathcal{W}$.*

Informally, the above definition requires that the system performs acceptably well for the behavior exhibited by the system when it is subjected to one of the specified input functions. The definition is, thus, in line with our informal definition stated above. Note that the definition is formulated in terms of the external behavior (i.e., B) visible at the system's interfaces rather than the internal behavior of the system. Thus, it treats a system as a black box; it is not important how the system internally achieves the desired adaptivity. Under this definition, *all* systems are adaptive with respect to some I, p, and $\mathcal{W}$. It is, thus, not important whether a system is adaptive but with respect to which I, p, and $\mathcal{W}$ it is adaptive. To simplify the discussion we omit p and $\mathcal{W}$ in the following, and we simply say that a system is adaptive with respect to I.

4 Self-Managing Systems

As stated in the previous section, an input function is vector-valued. In practice, some of the components of $i(t)$ represent environmental changes and disturbances, called *regular inputs*, while others represent external input applied to the system by a controller, called *control inputs*[1]. Intuitively, we expect from a self-managing system that it receives only regular input but no control input. This raises the question which inputs are to be called regular inputs and which are to be called control inputs. We answer this question analogous to Lendaris [6]: Which inputs are to be called regular inputs and which are to be called control inputs is determined by the performance function p because p evaluates certain inputs and the outputs of the system to derive its results. By definition, these inputs are the regular inputs, while all others are control inputs. Let $\mathcal{R}$ be the set of all regular inputs and let $\mathcal{C}$ be the set of all control inputs. Then, $\mathcal{I} = \mathcal{R} \cup \mathcal{C}$ holds for the system's input interface. A function $r(t) : \mathbb{T} \to \mathit{dom}(\mathcal{R})$ is called *regular input function*, where $\mathit{dom}(\mathcal{R})$ is the range of values of $\mathcal{R}$. A function $c(t) : \mathbb{T} \to \mathit{dom}(\mathcal{C})$ is called *control input function*, where $\mathit{dom}(\mathcal{C})$ is the range of values of $\mathcal{C}$. For a given $i(t)$, let $i_{\mathcal{R}}(t)$ and $i_{\mathcal{C}}(t)$ be the partial input functions that are derived by restricting i to those components which are regular inputs and those that are control inputs, respectively. For given $r(t)$ and $c(t)$, let $r(t) \circ c(t)$ be those $i(t)$ that arises by properly combining the components of $r(t)$ and those of $c(t)$ into a new vector.

According to its definition, an adaptive system is allowed to receive both, regular input and control input. In practice, those systems are of special interest whose control inputs can be computed solely from the system's past and current regular inputs and its past and current outputs. This leads us to the definition of self-manageable systems:

Definition 4 (Self-Manageable System). *A system* $\mathfrak{S} = (\mathcal{I}, \mathcal{O}, B)$ *with* $\mathcal{I} = \mathcal{R} \cup \mathcal{C}$ *is* self-manageable *wrt.* I *iff (1) it is adaptive wrt.* I *and (2) there exits a computable, nonanticipating behavior* $C : (\mathbb{T} \to \mathit{dom}(\mathcal{R}) \times \mathit{dom}(\mathcal{O})) \to (\mathbb{T} \to \mathit{dom}(\mathcal{C}))$ *such that:* $\forall (i \in I).(\exists i' \in I) : i' = i_{\mathcal{R}} \circ C(i_{\mathcal{R}}, B(i'))$.

We restrict the definition to nonanticipating control functions since a function anticipating future values of arbitrary input functions does not exist in practice.

A self-manageable system $\mathfrak{S}$ can be extended to a system $\mathfrak{S}'$ which consists of the original system $\mathfrak{S}$ and a component with behavior C acting as a *controller*

[1] Note that, for the purpose of this discussion, the environmental changes may also include purposeful input generated by humans, e.g., by the users of a software system.

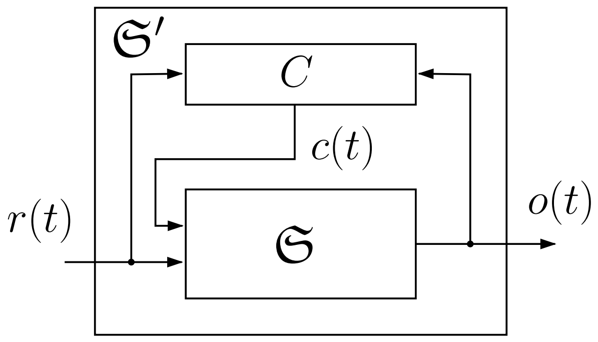

Fig. 1. Observer/controller pattern.

that computes the control input needed by $\mathfrak{S}$ to be adaptive from the regular input and the output of $\mathfrak{S}$. This resembles the well-known *observer/controller pattern* (Fig. 1). The resulting system $\mathfrak{S}'$ is adaptive and has no control input. Intuitively, such a system is self-managing. This leads us to the definition of self-managing systems:

Definition 5 (Self-Managing System). *A system $\mathfrak{S}$ is* self-managing *wrt. a set of input functions I iff it is adaptive wrt. I and $\mathcal{C} = \emptyset$.*

The above definition of self-managing systems is in line with our informal definition stated at the beginning of this section. We are aware that the separation of input into regular and control input needs further work.

5 Self-Organizing Systems

Intuitively, we expect from a self-organizing system that it adapts to its environment without outside control (i.e., that it is self-managing), and as the term "organization" implies that it establishes and maintains a certain kind of structure (cf. discussion in [3]). Moreover, we often associate robustness and scalability with self-organizing systems. As defined in the previous section, a black-box approach is sufficient to determine whether a system is self-managing or not. Detecting structure, however, is not possible within a black-box approach. Hence, we drop the black-box view and identify the components that constitute a system.

Definition 6 (Component). *A system $\mathfrak{S}$ consists of a (potentially empty) set of interacting components $\mathfrak{C}$, where each* component *$c \in \mathfrak{C}$ is a system itself, according to Def. 1.*

The system's components are organized in a certain way determining the structure of the system. They interact to provide the system's primary functionality, but their interactions may also change the structure of the system. In general, structures can be very diverse and occur in many dimensions. They may be spatial, temporal, spatiotemporal, or functional in nature and, thus, demand for a relatively abstract definition to cover their variety.

Definition 7 (Structure). Structure *is the property of a system* $\mathfrak{S}$ *by which it constrains the degrees of freedom of its components* $\mathfrak{C}$*.*

Often, structure can be described by a relation (or by a family of relations in more complex scenarios). For example, consider a packet-switched communication network which should setup and maintain a dedicated routing path between two terminal nodes to allow the terminals to communicate. In this case, the system's components are the network nodes and its structure is given by the established path which can be described by a binary relation containing all links (i.e., pairs of nodes) belonging to this path. A node is not free to forward a packet it receives on any of its outgoing links. Instead, it is restricted to forward it on that link leading to the next node on the path to the receiving terminal.

A common way to detect structuring is to apply the concept of information entropy introduced by Shannon [7]. This is done by inspecting the system's state space: When a system constrains the degrees of freedom of its components, it restricts its states to reside in a subset of the available state space. As a consequence, the entropy (uncertainty about the current state) of the system decreases indicating that the system gains structure. However, whether the entropy increases, decreases, or stays the same when observing a system depends largely on how the state space is defined [8].

A self-organizing system changes its structure in order to ensure its adaptivity. Using the concept of entropy, the evolution of a system's structuredness can be observed over time and it can be concluded that a system changes its structure if the degree of structuredness changes. However, the converse is not always true because a system can change its structure without affecting the degree of its structuredness. This leads to the following definition:

Definition 8 (Structure-adaptive System). *A system* $\mathfrak{S}$ *is* structure-adaptive*, iff (1) it is adaptive and (2) it adapts by dynamically changing its structure.*

The above definition combines Zadeh's notion of adaptivity with the system's structure stating that the system is adaptive because it dynamically changes its structure. Hence, to prove that a system is structure-adaptive it has to be shown that the system was not adaptive if it would not dynamically change its structure. Continuing our example from above, the communication system may change the path connecting the two terminals if a node that is part of the path fails. In this case, the system changes its structure to establish a new path that reconnects the two terminals. Clearly, this change is necessary to ensure that the terminals can communicate again.

Besides being structure-adaptive, self-organizing systems exhibit a further characteristic that distinguishes them from self-managing systems making them robust and scalable: decentralized control. If a system is controlled centrally, then it has a central point of failure. Accordingly, one way of proving that a

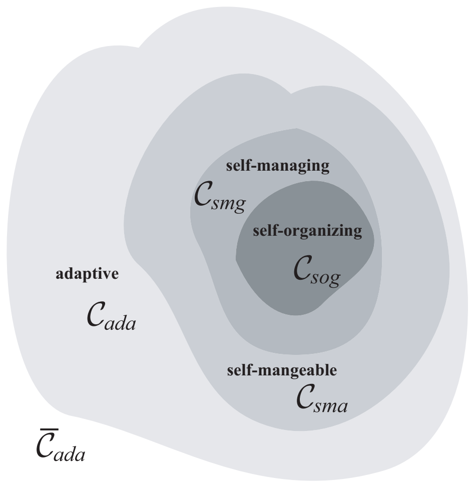

Fig. 2. Hierarchy of self-X properties.

system works in a decentralized fashion, is to show that it has no central point of failure [3]. Now, we are ready to define self-organizing systems:

Definition 9 (Self-Organizing System). *A system is* self-organizing *iff it is (i) self-managing, (ii) structure-adaptive, and (iii) employs decentralized control.*

The above definition of self-organizing systems suits our intuitive definition stated at the beginning of this section. It is also in line with our definition of self-organizing system previously given in [3]. We are aware that the definition of a decentralized control must be enhanced.

6 System Class Containment Relationship

Considering the various kinds of systems that have been defined in the previous sections (adaptive, self-manageable, self-managing, and self-organizing systems) it is apparent that there is a relation between them. Let us denote the set of adaptive systems with respect to a certain input space and performance function as $\mathcal{C}_{ada}$, the set of all self-manageable systems as $\mathcal{C}_{sma}$, the set of self-managing systems as $\mathcal{C}_{smg}$, and the set of all self-organizing systems as $\mathcal{C}_{sog}$ (each of them with respect to the same input functions, performance function, and acceptability criterion). Then, the following subset relationship holds (Fig. 2):

$$\mathcal{C}_{ada} \supseteq \mathcal{C}_{sma} \supseteq \mathcal{C}_{smg} \supseteq \mathcal{C}_{sog}$$

In the following section, we show that this hierarchical relation eases the classification of given systems.

7 Example Classification

In this section, we consider a set of exemplary systems $\mathfrak{S}_1, \ldots, \mathfrak{S}_4$ performing the same task. We want to examine to which class of our hierarchy each system belongs. Assume that $\mathfrak{S}_1, \ldots, \mathfrak{S}_4$ are networks that shall provide an *overlay path* between a specific input node N_i and a specific output node N_o. For this purpose, each node in the network can establish *overlay links* to nodes that are direct neighbors in the network. The graph of all potential links is called *underlay* graph. Each underlay link has a certain delay, and no node can support more than two overlay links at any time, except for N_i and N_o which only support one overlay link. Thus, if a new overlay link is established by a node that already has overlay links to two other nodes, one of these links has to be removed. The nodes of the network are impacted by failures. If a node is subject to a failure, it stops communicating. Moreover, nodes that have failed may be repaired. A repaired node starts communicating again. We define that $\mathfrak{S}_1, \ldots, \mathfrak{S}_4$ perform acceptably well, if there is no infinitely long service interruption (i.e., no overlay path between N_i and N_o) as long as the minimum failure inter-arrival time is above a given threshold T_l and as long as the underlay graph is not partitioned. Clearly, the occurrence of failures is a matter the system has to adapt to in order to perform acceptably well. Figure 3 depicts an underlay graph with overlay links.

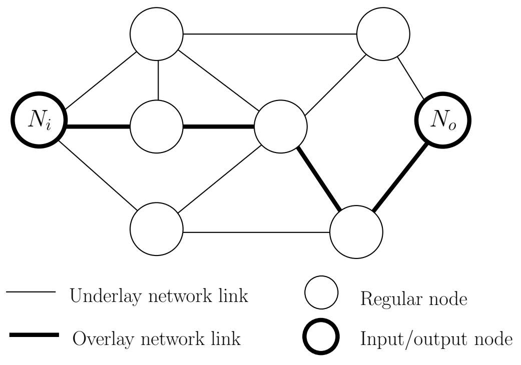

Fig. 3. Example underlay graph with overlay links.

The description given above applies to each of the systems $\mathfrak{S}_1, \ldots, \mathfrak{S}_4$. Now, we want to discuss the differences between these systems:

$\mathfrak{S}_1$: The system has a number of control inputs which control, which pairs of nodes establish an overlay link. For the delay d_c of the control inputs, the following holds: $d_c \gg T_l$.

$\mathfrak{S}_2$: The system differs from $\mathfrak{S}_1$ by a smaller delay of the control inputs. More precisely, the following holds: $d_c \ll T_l$.

$\mathfrak{S}_3$: In contrast to $\mathfrak{S}_1$ and $\mathfrak{S}_2$, the system has an additional component that collects the failure states of all nodes, calculates a valid path, and sends appropriate control inputs to the nodes. It has no external control inputs.

$\mathfrak{S}_4$: The system distributes the task of building a path between N_i and N_o among the nodes. To achieve this, an algorithm similar to the RIP protocol for routing [9] is used. We assume that the time needed by the distributed algorithm to fulfill its task is much shorter than T_l.

We now examine to which of our system classes each system belongs. First, we consider the subspace of the input that consists of the failure events. Since the performance function, which is identical for all systems, depends on the minimum failure inter-arrival time, the failure input is the regular input. We do not have to care about failure input sequences that partition the underlay graph or in which the time between failures is smaller than T_l, because our performance function allows any behavior in this case. The way all systems may adapt to failures is by altering overlay links. Clearly, by building an overlay path in the underlying graph, a *structure* is constituted. In addition to the failure input, the systems $\mathfrak{S}_1$ and $\mathfrak{S}_2$ have additional input, that has an impact on the behavior, but is not evaluated by the performance function. Thus, this input is control input.

For any failure input that does not partition the underlay graph, an appropriate control input may take care that a connection is always reestablished after a failure has occurred. Thus, both systems are adaptive with respect to the input space I consisting of the failures input and the proper control input. Hence, $\mathfrak{S}_1$ and $\mathfrak{S}_2$ belong to $\mathcal{C}_{ada}$.

For system $\mathfrak{S}_1$, the control inputs have a delay that is larger than the minimum failure inter-arrival time T_l. Hence, it is possible that a failure input pattern occurs that (1) disables one of the nodes that belongs to an existing overlay path and that (2) in the following always disables one of the nodes that are selected by the control inputs for re-establishing the path *before* the delay has passed. This would render the system into one that does not perform acceptably well since an overlay path between N_i and N_o will never be re-established. The *only* way to avoid this for sure is to indeed anticipate the failure signal. That means, the control signal cannot be computed from the past of the failure signal. Thus, $\mathfrak{S}_1$ does not belong to $\mathcal{C}_{sma}$.

The situation is different for $\mathfrak{S}_2$. If we assume that the computation of a new overlay path between N_i and N_o does take less time than $T_l - d_c$, a controller can take care that in case of a failure a new connection is established before the next failure occurs. Thus, $\mathfrak{S}_2$ belongs to $\mathcal{C}_{sma}$. However, since $\mathfrak{S}_2$ needs control input to perform acceptably well, it does not belong to $\mathcal{C}_{smg}$.

Since $\mathfrak{S}_3$ and $\mathfrak{S}_4$ do not need any control input to perform the same task as $\mathfrak{S}_1$ and $\mathfrak{S}_2$ acceptably well, they both belong to $\mathcal{C}_{smg}$ (and, thus, also to $\mathcal{C}_{sma}$ and $\mathcal{C}_{ada}$). We have already stated that all considered systems adapt by changing their structure. However, $\mathfrak{S}_3$ can be decomposed into two parts in such a way that removing one part lets lose the system's ability to adapt. That means, that it has a central controller and, thus, does not belong to $\mathcal{C}_{sog}$. $\mathfrak{S}_4$ has no central controller (cf. [3, Section 7.1]) and, thus, belongs to $\mathcal{C}_{sog}$. Figure 4 shows the resulting classification of the example systems.

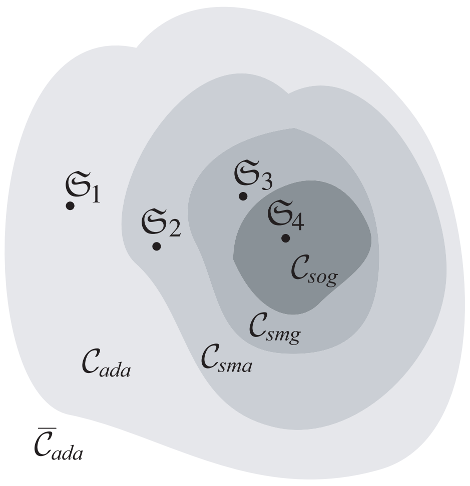

Fig. 4. Classification of example systems

8 Conclusions

In this paper, we presented a containment hierarchy which allows to categorize systems according to whether or not they are adaptive, self-manageable, self-managing, and self-organizing. We used Zadeh's [4] notion of adaptivity as a starting point and consecutively added further requirements to yield the other, more restrictive sets. Here, we applied the idea of Lendaris [6] to divide the input of the system into control input and regular input. This led us to self-manageable systems (whose control input can additionally be computed by a controller leading to the observer/controller pattern), self-managing systems (which additionally do not have any control input), and self-organizing systems (which additionally are structure-adaptive and employ decentralized control) for which we proposed formal definitions. We also showed that our classification is sensible by giving variants of an example system that fit into the different sets.

We hope that our proposal stimulates the further discussion about self-managing and self-organizing systems. For the discussion, we see the following open issues: Is the separation of the input into regular and control input always possible? What could be a precise definition of decentralized control? Despite of that, we believe that our approach is already of practical use. Moreover, we think that a similar clarification is also necessary for the other so-called self-X properties such as self-healing, self-configuring, self-optimizing, and self-protecting. This would path the way to a more scientific discussion of these terms.

References

1. Kephart JO and Chess DM: The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.
2. Herrmann K: *Self-Organizing Infrastructures for Ambient Services.* PhD thesis, Technische Universität Berlin, July 2006.
3. Herrmann K, Werner M, and Mühl G: A methodology for classifying self-organizing software systems. *International Transactions on Systems Science and Applications*, 2007. (accepted for publication).
4. Zadeh LA: On the definition of adaptivity. *Proceedings IEEE (Correspondence)*, 51:469–470, March 1963.
5. Willems JC: Paradigms and puzzles in the theory of dynamical systems. *IEEE Transactions on Automatic Control*, 36(3):259–294, March 1991.
6. Lendaris GG: On the definition of self-organizing systems. *Proceedings of the IEEE*, 52:324–325, 1964.
7. Shannon CE: A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656, July and October 1948.
8. Gershenson C and Heylighen F: When can we call a system self-organizing? *Advances in Artificial Life, 7th European Conference, ECAL 2003*, volume 2801 of *Lecture Notes in Computer Science*, pages 606–614, Dortmund, Germany, September 2003. Springer.
9. Hedrick CL: Routing Information Protocol. RFC 1058 (Historic), June 1988. Updated by RFCs 1388, 1723.