A Characterization of Key Properties of Environment-Mediated Multiagent Systems

Hartmut Schmeck[1] and Christian Müller-Schloer[2]

[1] Institute AIFB, Karlsruhe Institute of Technology (KIT), Germany
[2] Institute of Systems Engineering, Leibniz University Hannover, Germany

Abstract. The increasing presence of application scenarios which are based on large collections of active components having to adapt continuously to changing environmental requirements has led to several research initiatives with the objective to create new concepts for the design and operation of environment-mediated multiagent systems. In particular, Autonomic Computing (AC) and Organic Computing (OC) have developed the vision of systems possessing life-like properties: They self-organize, adapt to their dynamically changing environments, and establish other so-called self-x properties, like self-healing, self-configuration, self-optimization etc. The impact of these initiatives will depend crucially on our ability to demonstrate the benefits of these systems with respect to some essential properties. Therefore, we need a clear understanding of some key notions like adaptivity, robustness, flexibility, or their degree of autonomy, allowing for self-x properties.

In this paper, a system classification of robust, adaptable, and adaptive systems is presented. Furthermore, a degree of autonomy is characterized to be able to quantify how autonomously a system is working. The degree of autonomy distinguishes and measures external control which is exhibited directly by the user (*no autonomy*) from internal control of a system which might be fully controlled by an observer/controller architecture that is part of the system (*full autonomy*). Finally, learning and of trustworthiness are briefly addressed, since these are further essential aspects of self-organizing, adaptive systems.

1 Introduction

Some of the major challenges for systems engineering arise from the trend of increasing complexity in the design, development, and maintenance of technical systems and from the necessity to adapt continuously to changing environmental requirements. Organic Computing (OC), like other initiatives such as IBM's Autonomic Computing [1] or Proactive Computing [2], postulates the necessity of a paradigm shift in the design of future technical applications, see e. g. [3]:

"It is not the question whether self-organised and adaptive systems will arise but how they will be designed and controlled."

This emphasizes the inherent challenge to provide appropriate concepts and methods for dealing with intelligent, interacting systems, which might have to

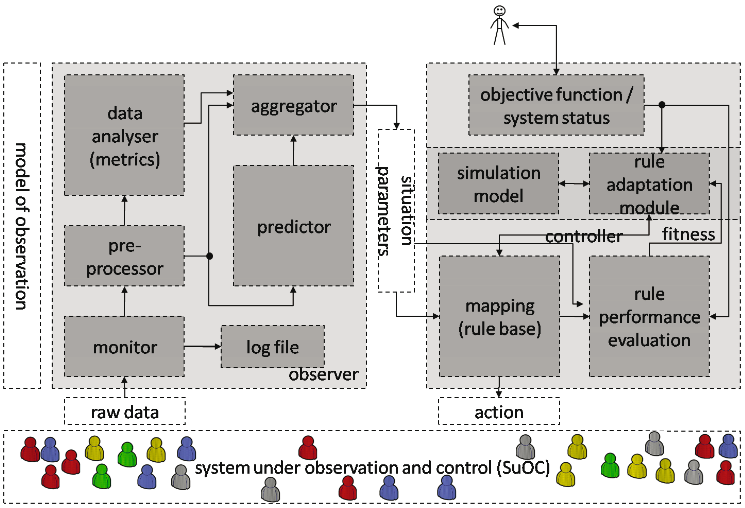


Fig. 1. Observer/controller architecture

cope with unanticipated events in their operating environment. In particular, we have to endow these systems with more degrees of freedom allowing them to self-organise in order to be able to *adapt* to potentially changing environmental conditions and external goals and constraints, and on the other hand we have to make sure that self-organization does not result in undesired behavior. Therefore, there is a need for concepts to achieve *controlled self-organization* as a new design paradigm. At first glance this seems to be a paradox, but it is necessary to cope in an acceptable way with the degrees of freedom required by the process of self-organization.

Within the German priority research program on OC [4] a generic architectural concept for the design and analysis of OC systems has been developed, the observer/controller architecture [5]. The (potentially) self-organizing *system under observation and control* (SuOC), which constitutes some productive system, will be endowed with a higher level of governance consisting of an *observer* and a *controller* (cf. Fig. 1). The observer monitors the underlying system by sampling the state and the properties of the different components and reports an aggregated quantified context (i. e. a description of the currently observed situation) to the controller. The controller evaluates this context with respect to a given objective function and takes appropriate control actions whenever it is necessary to influence the underlying system in order to meet the system goal. This loop of observing and controlling has to guarantee that the behavior of the SuOC stays within the external constraints. In particular, explicit control actions should not be necessary unless a deviation from the desired behavior has been detected or predicted, i. e. the SuOC would run autonomously as long as it behaves well and satisfies the requirements as specified by the developer or the user.

The feasibility of the control actions is evaluated continuously by comparing their predicted and observed impact. In this way, by on-line learning, every action gets an associated fitness value. The generic controller architecture provides an additional loop for off-line learning which consists of a simulation-based generation of new control actions having a minimal fitness level.

This generic observer/controller architecture bears similarities with concepts and techniques known from other scientific disciplines like control theory, mechanical engineering [6,7], or autonomic computing [1,8]. However, OC emphasizes the crucial difference from these disciplines that the resulting architecture is not a fully autonomous element. It is rather driven by external goals, and it reports its system status to the user (or to some higher level object). The user may intervene explicitly, e. g. by changing some system objectives or by initiating directly some control actions. Furthermore, the generic architecture allows for an adaptation of the control mechanism by utilizing on-line and off-line learning simultaneously.

An abstract view of the generic observer/controller loop is serving as our model for characterizing essential notions like robustness, adaptivity, or – ultimately – controlled self-organization in technical application systems. Our goal is to get significantly beyond the frequently followed attitude of using these terms in an almost magical sense based on some black box model. Despite of the long-term existence of these terms, in the context of technical applications based on advanced information processing systems we still lack a precise quantitative definition as a basis for a common understanding of these crucial concepts, which is a necessary prerequisite for a systematic comparison of different designs of self-organizing adaptive systems.

Based on a preliminary presentation at CEC 2007 [9], this paper elaborates on ideas towards this end that have been developed within the German priority research program on "Organic Computing" and it discusses some qualitative and quantitative characterizations of the essential properties of self-organizing technical systems. Section 2 introduces an abstract system model and provides an organic traffic light controller as an example of a self-organizing system. Characterizations of terms like robustness and adaptivity are given in Sect. 3, followed by a quantitative notion of autonomy and (controlled) self-organization. Aspects of learning and trustworthiness are briefly addressed in Sect. 5. Finally, Sect. 6 provides some concluding remarks.

2 System Description

Our characterization of terms uses a rather abstract description of a system which is based on the generic observer/controller architecture as depicted in Fig. 1. Let S be some productive system (the system under observation and control – SuOC), which processes some input and produces some output (cf. Fig. 2). The specific nature of the input and output is not essential for the purpose of this article, in a more detailed specification they would be specified as some parameter vectors.

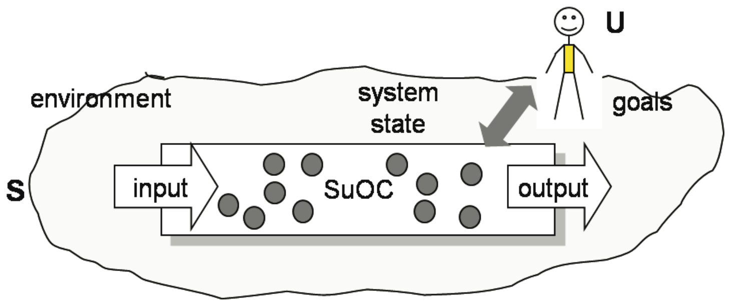


Fig. 2. System description

S is assumed to be a structured system, consisting of (many) elements and links, which interconnect these elements. We define the *system structure* to be an attributed graph, the nodes of which are the elements, and its edges correspond to the links between the elements. The *attributes* are associated to the system (*global attributes*) or to the edges and elements (*local attributes*) giving information like location, performance, storage capacity, communication bandwidth, reliability etc. A *re-organisation* of S adds or removes nodes and/or edges from the graph or modifies the attributes.

The behavior of the system may be influenced by its *environment*. This will subsume everything that is external to S, including input and output of S, and certain parameters (or attributes) which might have a disturbing effect on S.

At any given time t, the values of all the relevant attributes of the system constitute its *state* $\boldsymbol{z}(t)$, i. e. if there are n attributes used to describe the state of S, $\boldsymbol{z}(t)$ is a vector in some n-dimensional *state space* Z (also called *parameter space* of S).

These parameters include a description of the *evaluation criteria* (also called *objectives*) $\eta_1, \ldots, \eta_k$ which are provided by the (external) user U (or by some higher level entity). The evaluation criteria are assumed to map the system state into the set of real numbers. The evaluation of the system might involve rather complex operations, but this is irrelevant for the purpose of this paper. The criteria allow to define a hierarchy of subspaces of Z characterizing the performance of the system (cf. Fig. 3):

1. For simplicity, we assume that in an *ideal state* all the criteria evaluate to zero, i.e. their values are assumed to measure the deviation from an ideal state of the system. This set of ideal states is called *target space* TS. Typical examples of evaluation criteria could be (i) "The load of the worker nodes should be balanced.", (ii) "All the nodes with red colour should be in the upper half of the living space, the nodes with green colour should be in the lower half.", or (iii) "Move node x from A to B as fast as possible using not more than k units of fuel.". These examples show that the target space may be quite large (although, in some cases, it might be a unique single state $\boldsymbol{z}_\Omega$).

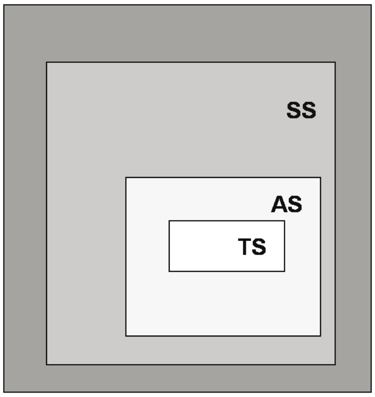


Fig. 3. State spaces of the SuOC

2. The system state $\boldsymbol{z}(t)$ is called *acceptable* if an *acceptance criterion* or *threshold* θ on $\mathbb{R}^k$ is satisfied, i. e. $\theta(\eta_1(\boldsymbol{z}(t)), \ldots, \eta_k(\boldsymbol{z}(t)))$ is true. As a special case, θ might refer to a vector of threshold values, which should not be violated (i. e. surpassed) by the evaluation criteria. The set of all acceptable states is also called the *acceptance space* AS. Obviously, the target space is a subset of the acceptance space. A typical example of such a threshold would be some upper bound on the accepted level of energy consumption while the ultimate objective would be to achieve an optimal value.
3. If the system is in an inacceptable state (i. e. $\boldsymbol{z}(t) \notin AS$), two cases are distinguished: If it is possible to modify the system state (by some control action), such that at some later time t' the state $\boldsymbol{z}(t')$ is acceptable, the system state is in the *survival space* SS. For example, if a car has a flat tire, it can return into an acceptable state again by replacing the flat tire with a spare tire. This might lead to reduced performance, i. e. the new state might be acceptable, but not ideal. Otherwise, $\boldsymbol{z}(t)$ is in the *"dead zone"*, i. e. the system cannot return into an acceptable state.

A crucial part of any system model is its dynamics, i. e. a specification of its state transitions. We assume that the potential state transitions are specified by some of the state attributes (e. g. a program or a transition table). Since we are explicitly interested in a distinction between internal and external control actions, we assume that there exist internal and/or external *control mechanisms* CM, which allow to control the behavior of the system by setting some attributes of the system and of its environment to specific values. Sometimes, it is assumed that it is not possible to control environmental parameters. But, as e. g. in traffic control, a speed limit could be viewed to be an environmental parameter which might be modified by some external control unit. As mentioned in Sect. 1, in the generic architecture of organic systems, CM will consist of an observer and a controller having a number of standard components as described in detail in

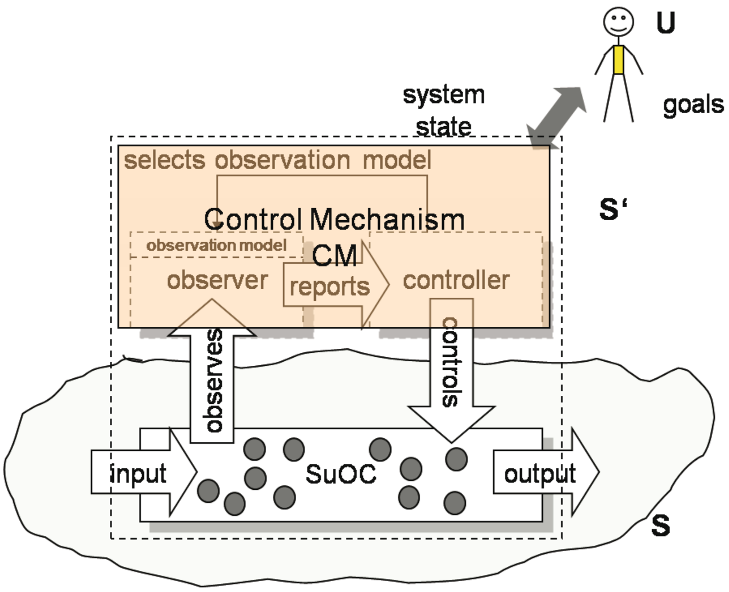


Fig. 4. System extended with control mechanism *CM*

[5]. The extended system model is sketched in Fig. 4; there, CM is external to S but internal to the extended system S'.

As a concrete example of such a system we refer to an organic traffic control system (OTC) as depicted in Fig. 5 (the example is taken from [10]). The system under observation and control is a traffic light controller which determines the length of the green periods for all the phases of a traffic light at some intersection. This could be a simple fixed-time controller or a more sophisticated traffic-adaptive controller which reacts to a number of different traffic parameters as indicated in Fig. 6. For OTC, the control mechanism consists of two levels:

- Layer 1 observes the current traffic situation (using appropriate sensors for detecting car frequencies, arrival rates etc.) and selects the most appropriate parameter settings for the traffic controller. This is done using a learning classifier system which is continuously evaluating the feasibility of its control actions by some kind of on-line learning.
- Layer 2 is triggered whenever there is no appropriate or insufficient response to a traffic situation: An evolutionary algorithm is used to generate more appropriate parameter settings which are evaluated off-line using a microscopic traffic simulator. Only those settings are transmitted to the lower level which show a sufficient performance. This is a crucial aspect of this application since it would not be acceptable to apply low quality control parameters in real traffic.

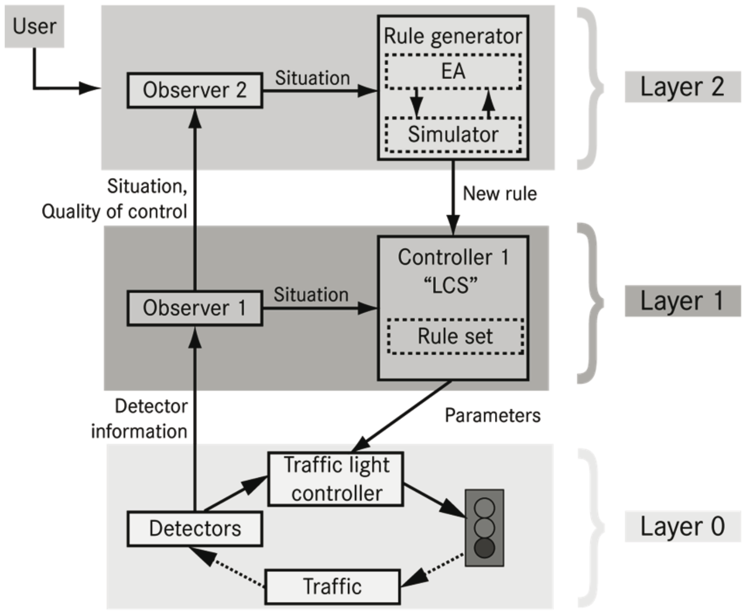


Fig. 5. Organic traffic control system (cf. [10])

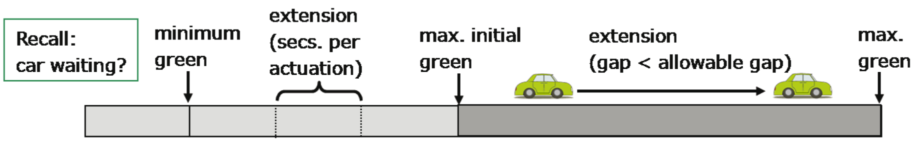


Fig. 6. Traffic-responsive traffic control (cf. [10])

For further details of the OTC architecture the reader is referred to [10]. The state vector of the OTC system consists of all the parameters that are relevant for its behavior: Traffic sensors, traffic light control parameters, elements of the rule base of the classifier system, the specification of the traffic simulator and the evolutionary algorithm. Obviously, this is a very long list of parameters. Therefore, the control mechanism has to select the most appropriate parameters by defining a *model of observation*, which is focusing the observer on the currently relevant attributes and selecting appropriate methods for analysis and prediction.

The following sections characterize crucial properties of self-organizing systems based on the system description outlined in this section.

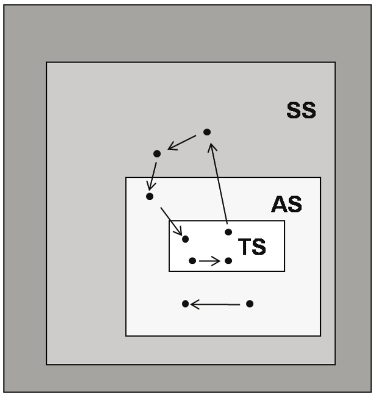


Fig. 7. State changes due to disturbances and control actions

3 Adaptivity, Robustness, and Flexibility

An essential requirement on system performance is the capability to adapt reasonably to changes in environmental parameters or in system objectives. In particular, a common objective is the capability to maintain a required behavior or functionality in spite of a certain range of parameter variations. The standard notion for this is *robustness*. Different from this, the requirement to modify the behavior because of certain changes of parameter values or of objectives would correspond to the notion of *flexibility*. Obviously, both notions crucially depend on the *adaptivity* of the system. In the following, a formalization of these concepts is presented.

The basic assumption underlying all these notions is the necessity of the system to react to changes of some parameter values. In our model, we are dealing with changes in the system state which might have very different origin, e. g. there might occur spontaneous changes of internal system parameters (e. g. due to component failures), or by environmental influences. Such a *disturbance* δ changes the state $\boldsymbol{z}(t)$ into some state $\delta(\boldsymbol{z}(t))$ (potentially with some time delay Δt and, consequently, the evaluation of the system changes from $\boldsymbol{\eta}$ $(= (\eta_1(\boldsymbol{z}(t)), \dots, \eta_k(\boldsymbol{z}(t))))$ to $\boldsymbol{\eta} + \boldsymbol{d}$ with $\eta_i(\boldsymbol{z}(t)) + d_i = \eta_i(\delta(\boldsymbol{z}(t)))$.

As shown in Fig. 7 the state changes due to disturbances can be differently severe with respect to the evaluation criteria. Some may leave the system within the target space whereas others might move it into the survival space such that it has to be "repaired" by some sequence of control actions. This leads to the following characterization of robustness:

Let D be a nonempty set of disturbances. A system S is called *strongly robust* with respect to D, iff all the disturbances $\delta \in D$ are mapping the target space and the acceptance space into themselves (i. e. they are mapping ideal states into ideal states and acceptable states into acceptable states).

Furthermore, a system S is called *weakly robust* with respect to D, iff all the disturbances $\delta \in D$ are mapping the acceptance space into itself (i. e. they might map ideal states into acceptable states, which would refer to a degradation in performance).

Obviously, strong robustness means that the system will continue to satisfy the evaluation criteria even under certain parameter changes, whereas weakly robust systems might show an acceptable deviation from an ideal behavior.

Example 1. An integrated circuit with automotive specification will function correctly from $-30\ ^{\circ}C$ to $+50\ ^{\circ}C$. Within this temperature range, there is no control action necessary, the system remains in an ideal state, and it is strongly robust with respect to changes in temperature that do not go beyond the specified range.

One could also define some quantitative notion of robustness by introducing a *degree of robustness*: The degree of robustness would grow with the size of D, and, if there exists some kind of distance measure on the state space, the degree of robustness would increase with the distances between the original and the disturbed state which do not lead to a significant deviation from ideal or acceptable states. In particular, a system S would be *more robust* than a system S', if it would be (strongly or weakly) robust with respect to more or stronger disturbances.

Using the distance measure on the state space, one could also characterize the robustness of S with respect to *smoothness*, i. e. small deviations in the parameter space should correspond to small(er) changes in the objective space.

By definition, if a system is in the survival space, there exists a sequence of control actions which move it back into the acceptance space. Therefore, we call a system *adaptable* with respect to a set of disturbances D, if the disturbances in D will not move the system out of the survival space. We could call it *perfectly adaptable* with respect to a set of disturbances D if for every state in the acceptance space and for every disturbance in D there is a sequence of control actions leading the system into an ideal state.

Now, while adaptability merely states that there is the potential to move a system back into the acceptance space or even into the target space, the much more interesting case would be the capability of a system to do so without any explicit external intervention. Consequently, a system S is called *adaptive* (with respect to some set of disturances D), if – after some time interval Δt – it returns into an acceptable (or even ideal) state from any disturbed state without needing any external control.

This definition of adaptivity does not refer explicitly to any internal control actions by some (central or decentral) control mechanism. This is an essential point, since there are many examples of self-organising systems where the adaptation to environmental requirements is an emergent effect of interactions among the components of the system. The standard biological example for such an adaptive system is the capability of ant colonies to find the shortest paths around obstacles that are placed or moved spontaneously in their environment (cf. [11]). This capability is due to stigmergic interaction between the ants, placing pheromones on their paths and preferring directions with higher pheromone concentrations when they are moving in their environment. Hence, the difference between adaptable and adaptive systems is not just a matter of setting system borders appropriately (i. e. placing control mechanisms outside or inside the system) but it addresses also the crucial emergent effects of interactions in self-organizing systems (cf. [12]).

As is obvious from the definition, every (weakly or strongly) robust system is also adaptive (since, in a robust system, the time interval Δt equals zero).

While these definitions are referring to the complete sets of ideal or acceptable states, one could define analogous notions of robustness or adaptivity with respect to single states only. This would allow to define different degrees of robustness for individual states. In an inverse approach, one could also define the notion of critical states:

A system S is in a *critical state* with respect to a set of disturbances D, iff every disturbance in D will transform the current state of S into an inacceptable state.

Although we did not specify how an adaptive system manages to move into an acceptable state, we assume that in order to achieve adaptable and/or adaptive behavior, control inputs $\boldsymbol{c}$ are triggered (such as switching on a cooling device or lowering the clock frequency) whenever S enters a state outside of its acceptance space (but inside its survival space, cf. Fig. 7), or whenever such a leave from its acceptance space is predicted. These control inputs (which sometimes are also called *control actions*) have to come from an external source (the *user*) in the case of an adaptable system and will be triggered by the internal control mechanism CM in the case of an adaptive system.

Hence, adaptive systems increase their effective robustness with the help of the internal control mechanism, which modifies the values of some parameters of the system or of the environment and thus influences the system behavior or structure. However, this might involve a temporary deviation from the acceptance space and thus may lead to a (temporary) decrease in system quality.

The *quality* of an adaptive system may be measured in different ways, some of which are listed below:

1. The distance of S from an ideal state (which may be defined as the norm of $\boldsymbol{\eta}(\boldsymbol{z}(t))$ in $\mathbb{R}^k$, e. g. a maximum norm), or, alternatively, the distance from an acceptable state (both refer to distances in the objective space, not in the parameter space),
2. The time it takes to move S from the present state in survival space back into the acceptance space (this could refer to real time or to the number of control actions that have to be applied),
3. The time it takes to move S into an ideal state, or
4. The maximum deviation of the system state from an ideal state (or the sum of all such deviations) on the path from an initial state in the survival space into the acceptance space.

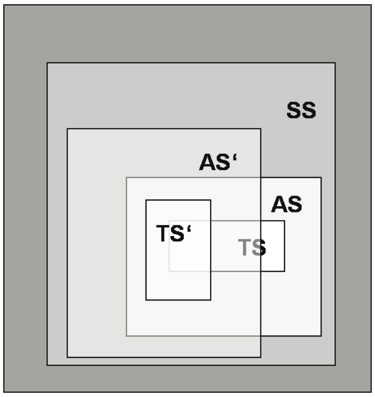


Fig. 8. Changes in state space structure due to changes in objectives

In general, we would like to distinguish two possible reasons for changes in the state of S:

1. The system state $\boldsymbol{z}(t)$ changes due to a change of the system (e. g. a broken component) or a change of the environment (disturbance δ) without any changes in the evaluation criteria. If the system remains in the acceptance space, this corresponds to the common understanding of a *robust system*.
2. The state $\boldsymbol{z}(t)$ changes due to modifications of the evaluation and acceptance criteria. This would modify the target space and the acceptance space (as indicated in Fig. 8). We call a system, which is able to cope with such changes in its behavioral specification, a *flexible system*.

After having defined these different types of adaptive behavior the next section focuses on a system classification with respect to different types of control mechanisms.

4 Degree of Autonomy and Controlled Self-organization

While in the previous section we have looked at whether the behavior of a system may be adjusted in response to changes in system or environmental attributes such that the acceptance criteria are eventually met, we now turn to the question to what extent this can be done by internal control actions only. That means we characterize essential properties of self-organizing systems.

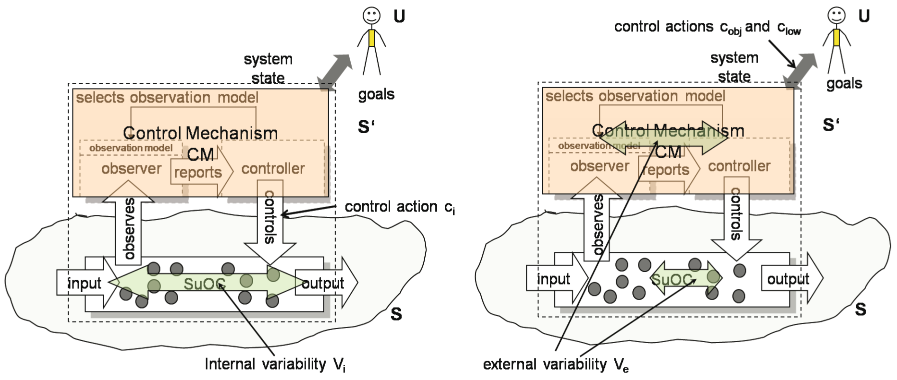
In our extended system model (cf. Fig. 4) we assumed the existence of some control mechanism CM that could be used to influence the system appropriately, if it deviates from the desired behavior. This requires adequate possibilities for modifications of system parameters, either by using external control (by the user, who might modify parameters of the CM) or by internal control which would modify parameters of the SuOC only. In the following we assume that S is an adaptable system.

The range of possible modifications is characterized as follows:

An *(internal) configuration* of the SuOC S is determined by the values of a collection of system or environmental attributes which are open to be modified by control actions of the CM (cf. Fig. 4). These attributes will also be called *configuration attributes*. The set of all the (theoretically) possible configurations constitutes the *(internal) configuration space* of S. The *variability* of the internal configuration space is measured by $V_i = \log(n)$ where n is the number of internal configurations.

Obviously, V_i corresponds to the number of bits necessary to address all the different configurations of the SuOC. The designer of a system has to specify explicitly which of the system and environmental attributes will be configuration attributes. Typically, this will include structural attributes of S. Furthermore, any evaluation and acceptance criteria that are used inside of S will typically belong to this set of attributes.

The control mechanism CM is responsible for selecting specific configurations of S. Therefore, every control action influences the values of a subset of the configuration attributes. We denote the number of bits of a specific control action $\boldsymbol{c}$ by $\#c$.



(a) internal configuration (b) external configuration

Fig. 9. Configuration spaces and control options

As mentioned already, the system S is assumed to be adaptable. Therefore, the control mechanism CM has the task to direct the system back into the acceptance space whenever a violation of the acceptance threshold is detected or predicted. Hence, a successful control mechanism will result in an adaptive system S' which is the combination of S and CM as indicated in Fig. 9. On the next level, the user (or some higher level entity) should also be able to influence the system. Therefore, there should be an adequate collection of *external configuration attributes*, defining the *external configuration space* which has an *external variablility* V_e (cf. Fig. 9(b)). These will be attributes of the control mechanism

CM, including always the evaluation and acceptance criteria and possibly some further attributes of this and the lower level SuOC S, in order to allow for direct influence of the *user* on *internal* attributes of the system S. Corresponding to these internal and external configuration spaces we distinguish between internal and external control actions $\boldsymbol{c}_i$ and $\boldsymbol{c}_e$, respectively.

An important design objective would be to reduce the variability of the configuration spaces going from the internal to the external level, i. e. in a multi-level design we should have an increasing degree of abstraction going from lower to higher levels. Nevertheless, it might be reasonable to provide direct higher-level control access to some configuration attributes of lower levels. Furthermore, in an adaptive system there should not be a necessity to specify anything more than evaluation and acceptance criteria by external control actions $\boldsymbol{c}_e$. Therefore, we distinguish between external control inputs $\boldsymbol{c}_{obj}$, specifying system objectives to be observed by the control mechanism CM, and control actions $\boldsymbol{c}_{low}$ directly related to attributes of the lower level SuOC. According to our definition of adaptive systems, there should not be a necessity for any control actions of type $\boldsymbol{c}_{low}$ as long as S remains within its survival space.

Using the definitions of internal and external variability, the *complexity reduction* from lower to upper level may be defined as $R = V_i - V_e$. R has a maximal value, if the external variability is zero, i. e. there is no externally visible variability which could be used for changing the configuration of the system. On the other hand, if the external configuration space includes attributes of the lower level system S, the external variability could even be larger than V_i. In such a case, the configuration task of the external user has even larger complexity than the configuration task of the control mechanism CM and, consequently, the complexity reduction R would be negative.

Now, these notions can be used to characterize the degree of autonomy of the extended system S'. Intuitively, the autonomy of a system should increase with a decrease in external interventions and vice versa. Therefore, the relationship between the variabilities V_i and V_e should be an intuitively reasonable indicator of the degree of autonomy. This leads to the following definition:

The *(static) degree of autonomy* α of system S' is defined as

$$\alpha = \frac{R}{V_i} = \frac{V_i - V_e}{V_i} \tag{1}$$

The value of α will be at most one (if V_e equals zero). In this case, there is no external variability, i. e. there is no possibility to modify any attributes of S' by external control actions. Therefore, such a system would be called *fully autonomous*. The only way to interact with such a system would be to influence the input values of S.

If the value of α is zero, the internal and the external variabilities are the same, which indicates that there is no reduction in complexity. If α drops below zero, we have a situation where the external configuration space contains more controllable attributes than the internal configuration space. In an extreme case, all the lower level attributes could be available externally, in addition to the configuration attributes referring to the control mechanism CM. In this case, the system S could be controlled completely by external control actions, which would be no autonomy at all. But, as is obvious from the definition, a low value of α is merely indicating a *potential* loss of autonomy, since it does not take into account, whether during any time interval any external control actions have been used actually. That means, even in a system with a rather low value of α the actual behavior might be more autonomous than indicated by α.

Referring to Fig. 7 and Fig. 9 again, CM influences (controls) S to keep it within its acceptance space by selecting a sequence of control actions $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots$ from its *behavioral repertoire* B. CM determines these control actions with respect to the evaluation and acceptance criteria as given initially or some time before by higher-level control. Ideally, it should be able to keep S within the acceptance space without additional intervention from a higher level. However, if the acceptance criterion is violated for too long, the higher level controller might intervene by sending further external control actions of type $\boldsymbol{c}_{obj}$ to modify the evaluation criteria or of type $\boldsymbol{c}_{low}$ to influence directly some of the configuration attributes of the SuOC.

Therefore, in order to characterize the actual degree of autonomy (or, the actual amount of external control) one should consider the number of bits that have been used in any control actions over some time period $[t_1, t_2]$. This leads to a more refined, time dependent quantitative notion of autonomy:

Let $v_e(t)$ and $v_i(t)$ be the number of bits of the external and internal control actions at time t, respectively.

(a) The *dynamic complexity reduction* r in S' over some time interval $[t_1, t_2]$ is defined to be

$$r_{t_1,t_2} = \int_{t_1}^{t_2} (v_i(t) - v_e(t))dt \tag{2}$$

(b) The *dynamic degree of autonomy* β in S' over some time interval $[t_1, t_2]$ is defined to be

$$\beta_{t_1,t_2} = \frac{\int_{t_1}^{t_2} (v_i(t) - v_e(t))dt}{\int_{t_1}^{t_2} v_i(t)dt} \tag{3}$$

This dynamic degree of autonomy measures the relative amount of external control that has been used during a specific time interval. Hence, a system could exhibit a fully autonomous behavior over some time period while being completely influenced by external control actions at a different time. This definitely allows for a much more accurate characterization of the autonomy of a system, based on the actual control flow. In particular, in the case of organic systems, we might have a fully autonomous behavior over some time period, but might use adequate control actions whenever the self-organized behavior of the system does not satisfy the external constraints.

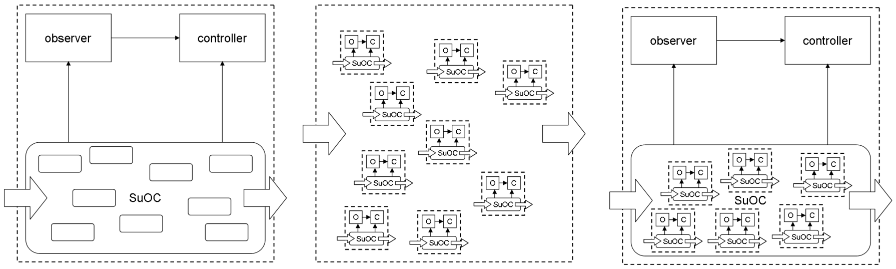
After having characterized the (dynamic) degree of autonomy of a system, we may start to look at the so-called *self-x-properties*, as the degree of autonomy would correspond to the *degree of self*. One of the central notions is "self-organization", a term which has attracted growing attention by scientists from different diciplines. F. Heylighen has presented valuable surveys on the relevant literature (see e. g. [12]), and he provided concise specifications of a range of relevant terms (for a very recent publication see [13]).

Heylighen states: "Self-organization can be defined as the spontaneous emergence of global structure out of local interactions." And, in the FAQ list on self-organized systems [14], we find "The essence of self-organization is that system structure often appears without explicit pressure or involvement from outside the system.".

Following these characterizations, a self-organizing system is a multi-element system (consisting of m elements, $m > 1$), which needs no external control to restructure itself (i. e. it has a high degree of autonomy). Furthermore, a common assumption is that the internal control mechanism CM is distributed over the elements. In natural systems, such a distributed control mechanism might be difficult to localize which, sometimes, leads to almost magical connotations of self-organization.

Obviously, in a controllable technical system there must always be some control mechanism. It can be centralized (one CM), distributed over the m elements (m CMs) or distributed over a hierarchy of CMs (possibly more then m CMs). These architectural variants are depicted in Fig. 10). Following a recent discussion, the centralized variant could also be viewed to address only the aspect of adaptivity, and therefore correponds to top-down design, whereas the decentralized variant resembles self-organized systems, where the global behavior is rather developed bottom-up (as an emergent process). Finally, the multilevel design combines the two design approaches and corresponds to an intuitive idea of controlled self-organization.

We could define a *degree of self-organization* by counting the number of CMs ($= k$) in relation to the number of elements m of the system ($k : m$). An adaptive system with one centralized CM could then be denoted as ($1 : m$), an adaptive system with full distribution of CMs as ($m : m$). Definitely, an ($m : m$) system may be called self-organized. It is a matter of taste whether we want to call a system with lower degrees of self-organization still self-organized.



(a) central (b) decentral (c) multi-level

Fig. 10. Observer/controller realization

Let S be an adaptive system consisting of m elements ($m > 1$) with a large dynamic degree of autonomy (β) and fully or partially distributed k control mechanisms CM ($k \geq 1$) leading to a degree of self-organization of ($k : m$).

(a) S is called *strongly self-organized*, if $k = m$, i. e. the degree of self-organization is ($m : m$).
(b) S is called *self-organized*, if $k > 1$, i. e. it has a medium degree of self-organisation ($k : m$).
(c) S is called *weakly self-organized*, if $k = 1$, i. e. there is a central control mechanism and the degree of self-organization is ($1 : m$).

The weak definition would mean that any adaptive system is also called self-organized. The authors of this article tend to use the medium definition. Speaking about adaptive and self-organizing systems in an exact way requires to indicate (1) the degree of autonomy and (2) the degree of self-organization ($k : m$).

Here, a remark seems to be necessary: In the literature about self-organizing systems (see e. g. [15]) you sometimes find the additional requirement that a self-organizing system should be *structure-adaptive*. This is not explicitly stated in our definition, but it may be present in an implicit form: For example, if the system objectives refer to structural configuration attributes of S then, after some disturbance or after a change in the evaluation and acceptance criteria, an adaptive system will be able to re-organize its structure such that it is acceptable again.

A common way of characterizing changes in system structure would consider the resulting changes of the system's descriptional complexity or of the entropy of appropriate structural attributes of S (e. g. by computing the classical information theoretic entropy as defined by Shannon). This has been done in a systematic way in [16], it is beyond the scope of this paper.

Finally, we suggest the following characterization of controlled self-organization:

A self-organized system S' allows for *controlled self-organization* iff

(i) it has a nonempty external configuration space, i. e. we have $V_e > 0$ (essentially we require that it has a small static degree of autonomy and allows for detailed information on its current state),
(ii) it has some external control actions of type $\boldsymbol{c}_{obj}$,
(iii) it has some external control actions of type $\boldsymbol{c}_{low}$, and
(iv) it has a large dynamic degree of autonomy (i. e. external control actions are rare).

This characterization requires that the system is adaptive (which means that it is capable of returning autonomously into an acceptable state after certain disturbing events), but provides the option to the external user to redefine certain objectives (by using a control input $\boldsymbol{c}_{obj}$) or to interfere directly with the operation of system S by using a control action of type $\boldsymbol{c}_{low}$. The simplest example of the latter would be a control action to switch off the system.

5 Learning and Trustworthiness

A major motivation for the interest in environment-mediated multiagent systems is the expectation that they would be capable to overcome certain limitations caused by incomplete knowledge about the potential operation environment at design time.
In particular, the behavior of S' may be limited by

(a) a limited configuration space of S or
(b) a limited behavioral repertoire B of CM.

Limitation (a) may be caused by an inappropriate design, i. e. the necessity for including additional attributes of the system or of the environment into the configuration space has not been seen at design time. There might also be some system specific reasons that prohibit any external modifying access to certain structural attributes of the system. In these cases, it does not seem reasonable to expect a possibility to extend the configuration space at runtime of the system.

Limitation (b), however, might occur quite often in systems having a very large configuration space but only a limited behavioral repertoire. In particular, it might be a complex optimization problem to find the most suitable configuration for satisfying the system objectives, as e. g. in traffic light control, where you have to determine the most appropriate values for a number of control parameters of an adaptive controller system (cf. [10]). In this example, the quality of a particular parameter setting will be determined either off-line by a simulation or on-line by observing and analysing the system behavior. Whenever the control mechanism CM or the user detect that the currently available range of control actions is not sufficient to get acceptable behavior, it should be possible to extend the behavioral repertoire with new, more appropriate control actions. Definitely, it would be desirable to have a system that is capable of generating the necessary modifications of its behavioral repertoire in an autonomous way. This will be addressed in the following subsection.

5.1 Learning

In this subsection we briefly address the aspect of learning in adaptive and self-organizing systems.

In general, a system has the capability to learn, if it can improve autonomously its response to input values from some set X. That means, there are time values t_1 and $t_2 > t_1$ such that for any $t > 0$ the response to an input from the set X at time $t_2 + t$ has a higher quality than the response to the same input at time t_1. Consequently, the robustness and performance of a (self-)learning system should improve over time.

This learning capability requires some *learning mechanism LM* which may modify the behavior of the system S' by

(a) changing the values of some attributes of the system S or of its environment, or
(b) changing the behavioral repertoire B of the control mechanism CM.

An interesting mechanism for learning by modifying parameters of the environment is the stigmergic use of pheromones by ant colonies. This aggregation of individual experiences combined with some degree of evaporation leads to the amazing capability of constructing shortest path ant roads even in a dynamically changing topography. This has inspired a whole range of new design patterns for optimization algorithms (cf. [17]).

The design of the organic traffic control system may serve as an example for the second type of learning method (cf. [10] and Sect. 2): A classifier system for selecting parameter settings for a traffic light controller (in the real traffic system S) is using on-line learning by associating a fitness value to classifier rules based on the performance of their parameter settings in real traffic situations, combined with off-line learning which produces new classifier rules for inadequately handled traffic situations by using a genetic algorithm which generates new rules of some minimum quality level by evaluating their performance in a traffic simulator. In this way, the learning mechanism manages to improve the system performance on known traffic situations and it is also capable of generating adequate responses (i. e. control actions) to previously unknown traffic situations.

These are just two examples of a broad range of possible learning mechanisms, which could make use of learning by experience, trial-and-error, reinforcement learning, neural networks, or metaheuristics like genetic algorithms, ant colony optimisation, or simulated annealing, to name a few.

In highly complex systems, autonomous learning is the most attractive way of coping with the limitations of adaptivity as mentioned in the previous subsection.

5.2 Trustworthiness

Another important feature of systems that can adapt to their environmental requirements and that can cope with unanticipated situations in a self-organized way is their trustworthiness. That means, in spite of the flexibility and adaptiveness of the system, a user will ask, why or how she should build up trust in the behavior of such a system? Simultaneously, a designer or service provider should know what kind of methodology is appropriate to generate and comprehensibly demonstrate and maintain trust in a system or service (potentially operating in an insecure environment). Without adequate levels of trustworthiness we cannot expect sufficient acceptance of these systems!

Obviously, there are many facets characterizing trustworthiness. In particular, the following properties address essential aspects of trust:

- *Correctness*: Does the system actually do what it should do?
- *Security*: Does the system prevent any unauthorised access?
- *Safety*: Will there be any undesired effects by using the system or service?
- *Availability / Reliability*: What is the probability that a service is available when it is needed and what is the expected time duration of satisfactory service?
- *Robustness*: Will the service be provided even within an environment that is changing due to some disturbances?

– *Privacy*: Will the service use privacy information in an adequate way?
– *Performance*: Will the system show the expected response time or throughput?
  Sometimes, trust is reduced to CIA:
– *Confidentiality*
– *Integrity*
– *Availability* to authorized users.

The final three properties all relate to security, they are quite often summarized as "Trusted Computing". But, trustworthiness extends far beyond these security-properties.

Some characterization has been given by S. Yau in his keynote talk at ATC 2006 [18] :

– Trust is a particular level of belief of an agent (the *trustor*) that some other agent (the *trustee*) will act or intend to act beneficially.
– Trust forms the basis for agents to make decisions on *what* to interact with, *when* to interact with, and *how* to interact with.

Consequently, trust management has to consider the (dynamic) relationship between trustor and trustee. The provision of *certificates* containing statements on the expected behavior of a system or service may be necessary for the establishment of trust, but they do not suffice. In particular, trust is not a binary property, there will always be a certain, dynamically changing level of trust only. In our characterization of adaptable and adaptive systems we emphasized their capability to return to satisfying or even to ideal behavior after the occurence of failures. But, the level of trust will be reduced whenever a system shows unacceptable behavior, and it will be more difficult to regain trust after the system returned to its acceptance space again. Another mechanism besides certificates is to use *reputations* for sharing experiences on system usage and performance. In some way, these reputations have a similar impact as the pheromones in ant systems: They may be used as an additional source of accumulated information on the previous behavior of a system.

As emphasized in this paper, we are interested in robustness and adaptivity because of potential deviations in system behavior from an ideal state. Consequently, trust management is closely related to risk management. Self-organized, adaptive services will always bear the risk of unexpected behavior and even a well-engineered system may run into situations where it does not "behave well". Therefore, the more we know about the risks associated with using a system, the more we will be prepared to cope with failures. It is important to note that information about the potential risks of a system provides the basis for the design of methods for preventing the occurrence of risky behavior and for building up trust. Hence, in order to fully appreciate the benefits of adaptive, self-organized behavior, it would be detrimental to ignore potential risks.

These few statements underline the necessity to develop a methodology for trust engineering for self-organizing environment-mediated systems which, among others, has to address the following:

– How do we build up initial trust?
– How can a trustor or trustee regain trust after failures?
– What can a producer do to create an initial level of trust for his product and to support a reasonable trust management?

Currently, we are still far away from satisfying answers to these questions, which might allow for coping with dynamic trust-relationships.

6 Conclusion

This contribution has given an overview on some recent characterization of important concepts and properties of self-organising and adaptive systems which are closely related to environment-mediated multiagent systems. Considering the increasing presence of interacting systems in our living environment, it is essential to design systems in a way which enables them to cope with unanticipated situations in a reasonable way and which – in particular – provides them with the capability to react appropriately to the frequently rather spontaneous and unexpected behavior of human users.

A common understanding of basic properties of these systems is a prerequisite for making reliable statements about their (expected) performance. Furthermore, as system designers and users, we have to know about the requirements for building up trustworthy relationships between services or systems and their users. We are beginning to be overloaded with undesired, unexpected, and sometimes even malicious information services. Therefore, we are in urgent need of systematic approaches to circumvent these problems and provide acceptable systems, which will respond as requested to environmental requirements.

The challenge for system design lies in building systems,

– which meet their target performance (stay within the target space)
– with little external control effort (high dynamic degree of autonomy)
– for a large set of environmental conditions and possible disturbances (large survival space).

It is essential for such a system to have a built-in control mechanism (constituting the *self*). A system will become more robust if the control mechanism is distributed, i. e. its degree of self-organization ($k : m$) is high since this avoids a single point of failure.

Adaptive and self-organising systems will not always outperform conventional systems but they may survive under a large diversity of environmental conditions. Self-organising systems may be especially robust due to their internal distributed architecture and their large structural variability.

The objective for system design should be to design controllable self-organising systems, i. e. systems, which allow for external control but have a high degree of autonomy as well. One major argument for the possibility of external control is the necessity to support the generation of trust into the dependability of a system.

Acknowledgement. This paper is based on intensive discussions and joint work with the members of our research groups at the Leibniz Universität Hannover and at the Karlsruhe Institute of Technology, we are deeply indebted to their valuable feedback and constructive suggestions. Furthermore, we gratefully acknowledge the support provided by the German Research Foundation (DFG) within the priority program on "Organic Computing".

References

1. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. IEEE Computer 1, 41–50 (2003)
2. Tennenhouse, D.: Proactive computing. Communications of the ACM 43, 43–50 (2000)
3. Schmeck, H.: Organic Computing – A new vision for distributed embedded systems. In: Proceedings of the 8th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2005), pp. 201–203. IEEE Computer Society Press, Los Alamitos (2005)
4. DFG Priority Program 1183 Organic Computing (2005) (visited June 2007), `http://www.organic-computing.de/SPP`
5. Branke, J., Mnif, M., Müller-Schloer, C., Prothmann, H., Richter, U., Rochner, F., Schmeck, H.: Organic Computing – Addressing complexity by controlled self-organization. In: Margaria, T., Philippou, A., Steffen, B. (eds.) Proceedings of the 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2006), Paphos, Cyprus, pp. 200–206 (2006)
6. Oberschelp, O., Hestermeyer, T., Kleinjohann, B., Kleinjohann, L.: Design of self-optimizing agent-based controllers. In: Urban, C. (ed.) Proceedings of the 3rd International Workshop on Agent Based Simulation, Passau, Germany. SCS European Publishing House (2002)
7. Hestermeyer, T., Oberschelp, O., Giese, H.: Structured information processing for self-optimizing mechatronic systems. In: Araújo, H., Vieira, A., Braz, J., Encarnação, B., Carvalho, M. (eds.) Proceedings of the 1st International Conference on Informatics in Control, Automation and Robotics (ICINCO 2004), pp. 230–237. IEEE Computer Society Press, Los Alamitos (2004)
8. Sterritt, R.: Autonomic Computing. Innovations in systems and software engineering 1, 79–88 (2005)
9. Cakar, E., Mnif, M., Müller-Schloer, C., Richter, U., Schmeck, H.: Towards a Quantitative Notion of Self-organisation. In: Proceedings of the 2007 IEEE Congress on Evolutionary Computation, pp. 4222–4229 (2007)
10. Rochner, F., Prothmann, H., Branke, J., Müller-Schloer, C., Schmeck, H.: An organic architecture for traffic light controllers. In: Hochberger, C., Liskowsky, R. (eds.) Informatik 2006 – Informatik für Menschen. LNI, vol. P-93, pp. 120–127. Köllen Verlag (2006)
11. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: Optimization by a colony of cooperating agents. Technical Report 26 1. IEEE Transactions on Systems (1996)
12. Heylighen, F.: The science of self-organization and adaptivity. In: The Encyclopedia of Life Support Systems, pp. 253–280 (1999)
13. Heylighen, F.: Complexity and self-organization. In: Bates, M.J., Maack, M.N. (eds.) Encyclopedia of Library and Information Sciences. Taylor & Francis, Abington (2008)

14. Lucas, C.: Self-organizing systems (sos) faq. Frequently asked questions version 2.99 (2006) (visited June 2007), `http://www.calresco.org/sos/sosfaq.htm`
15. Mühl, G., Werner, M., Jaeger, M.A., Herrmann, K., Parzyjegla, H.: On the definitions of self-managing and self-organizing systems. In: Braun, T., Carle, G., Stiller, B. (eds.) Proceedings of the KiVS 2007 Workshop: Selbstorganisierende, Adaptive, Kontextsensitive verteilte Systeme (SAKS 2007), Bern, Switzerland, pp. 291–301. VDE Verlag (2007)
16. Müller-Schloer, C., Sick, B.: Emergence in Organic Computing systems: Discussion of a controversial concept. In: Yang, L.T., Jin, H., Ma, J., Ungerer, T. (eds.) ATC 2006. LNCS, vol. 4158, pp. 1–16. Springer, Heidelberg (2006)
17. Dorigo, M., Stützle, T.: Ant colony optimization. B&T, MIT Press (2004)
18. Yau, S.S.: Managing trust in distributed agent systems (keynote address). In: Proceedings of 3rd International Conference on Autonomic and Trusted Computing (ATC), pp. 17–25 (2006)