Electronic Communications of the EASST
Volume 27 (2010)

Workshop über Selbstorganisierende, adaptive, kontextsensitive verteilte Systeme (SAKS 2010)

Modeling of Self-Organizing Systems: An Overview

Richard Holzer, Patrick Wüchner, Hermann de Meer

12 pages

Guest Editors: Klaus David, Michael Zapf
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Modeling of Self-Organizing Systems: An Overview

Richard Holzer[1], **Patrick Wüchner**[2], **Hermann de Meer**[3]*

[1]University of Passau, Germany, holzer@uni-passau.de
[2]University of Passau, Germany, patrick.wuechner@uni-passau.de
[3]University of Passau, Germany, hermann.demeer@uni-passau.de

Abstract: This paper gives a systematic overview on modeling formalisms suitable for modeling self-organizing systems. We distinguish between micro-level modeling and macro-level modeling. On the micro level, the behavior of each entity and the interaction between different object must be described by the model. Macro-level modeling abstracts from the individual entities and only looks at the behavior of the system variables of interest. The differentiations between discrete and continuous time and between discrete and continuous state space lead to different descriptions of the model.

Keywords: Self-Organization, Modeling, Systems

1 Introduction

One main goal for networking systems is to reduce administrative requirements for users and operators. The system should be able to manage and configure itself as much as possible without requiring human effort. A self-organizing system should also be able to detect and correct failures automatically if possible. The concept of self-organization is a topic that has become more and more important in the last few years. A lot of research effort has been done regarding the design and analysis of self-organizing systems. To assist these technological advances, mathematical models foster the understanding of complex systems and facilitate the design of new systems. The main goal of this paper is to contribute to this issue.

This paper gives an overview on modeling methods for self-organizing systems. According to [DK05] and [Hey03], typical features of self-organization are adaptivity, autonomy, emergence, decentralization, and self-maintenance. Therefore, the focus of this paper lies on the modeling of systems consisting of many entities interacting with each other to fulfill a global goal without any central control. We distinguish between macro-level modeling and micro-level modeling. While during micro-level modeling the behavior of each entity of the system must be described, macro-level modeling uses the technique of aggregation to derive a model for the system variables of interest. Each macro-state can be seen as an equivalence class of micro-states. For both modeling methods, we have to decide which parts of the system are modeled discrete and which parts are modeled continuous. Another classification is the determinism: If some entities of a real-world system are too complex to be modeled in all details, the behavior of the entities may be modeled

* This research is partially supported by the SOCIONICAL project (IP, FP7 Call 3, ICT-2007-3-231288), by the ResumeNet project (STREP, FP7 Call 2, ICT-2007-2-224619) and by the Network of Excellence EuroNF (IST, FP7, ICT-2007-1-216366).

as non-deterministic by specifying probability distributions. This leads to stochastic automata on the micro-level or to stochastic processes on the macro-level.

The applicability of different approaches to real systems is demonstrated in, e.g., [Boc04], [HDB08], [HD08], [BGDT06], [AWD08], and [GSB02]. We do not consider the topic of robust control theory due to lack of space. The interested reader is referred to [ZD97].

2 Macro-Level Modeling

For macro-level modeling, the global state space of a system is reduced to the *relevant* properties, while selecting these properties is one of the most crucial decisions the modeler has to take. For defining the macro-level model, the following items of the model have to be specified:

- Dynamic variables $x_t = (x_t^{(1)}, x_t^{(2)}, \ldots, x_t^{(n)})$ which change their values during the time t. They are usually the variables of interest that we would like to analyze. However, the variables of interest might depend on additional variables. For example, all relevant inputs and outputs of the system are modeled as dynamic variables.
- Macro-state space S which is a subset of the Cartesian product $S \subseteq S_1 \times S_2 \times, \ldots S_n$, where S_i is the set of all possible values of the dynamic variable $x_t^{(i)}$ for $i = 1, 2, \ldots, n$.
- Set $T \subseteq \mathbb{R}$ of points $t \in T$ in time at which a macro-state change of the system can be observed. For all other points $t \in \mathbb{R} \setminus T$ in time, there will be no observable macro-state change. Macro-state changes of the system may occur discretely (e.g., $T \subseteq \mathbb{N}$) or continuously (e.g., $T \subseteq \mathbb{R}_0^+$). If the behavior of the system with respect to an initial state is analyzed, time $t = 0$ usually refers to the initial state, i.e., $\min T = 0$.
- Static system parameters $p = (p_1, p_2, \ldots, p_k)$: Unlike dynamic variables, these parameters do not change over time, but they may influence the change of dynamic variables.
- Behavior rules R for the dynamic variables that describe the (deterministic or stochastic) change of the dynamic variables x_t in dependency of the static system parameters p.

After specifying these items, we are able to analyze the behavior of the whole system: For each initial macro-state $x_0 \in S$, we can apply the rules R to derive all macro-states of the system as a function depending on the time t: $(x_t)_{t \in T}$. This mapping is called the *orbit* (also known as *sample path* or *state trajectory* [CL08, p. 14]) of the system with respect to the initial value x_0. In a deterministic system, the orbit is uniquely determined by the initial macro-state x_0. In stochastic systems, the orbit depends on random events.

An equilibrium [Boc04] is a macro-state $x^* \in S$ such that a system starting in $x_0 = x^*$ will not leave this macro-state, i.e., the orbit of this system contains only one macro-state x^*. The equilibrium x^* is stable, if a system starting near the equilibrium x^* will stay near x^*, i.e., for all $\varepsilon > 0$, there exists $0 < \delta \leq \varepsilon$ such that, for each initial macro-state x_0 with $d(x_0, x^*) < \delta$, we get $d(x_t, x^*) < \varepsilon$ for all $t \in T$, where $d : S^2 \to \mathbb{R}_0^+$ is a metric (distance function) on the macro-state space S (e.g., $d(x, y) = |x - y|$ in the Euclidean space $\mathbb{R}^n$). A stable equilibrium x^* is asymptotically stable, if a system starting near the equilibrium x^* will converge to x^*, i.e., there exists $\delta > 0$ such that, for each initial macro-state x_0 with $d(x_0, x^*) < \delta$, we get $\lim_{t \to \infty} x_t = x^*$.

In the following sections, we distinguish between discrete-time and continuous-time macro-level modeling. Moreover, also the macro-state space can be chosen as discrete or continuous. Figure 1 illustrates these interrelationships (cp. Figs. 24.1 and 24.2 of [Jai91]).

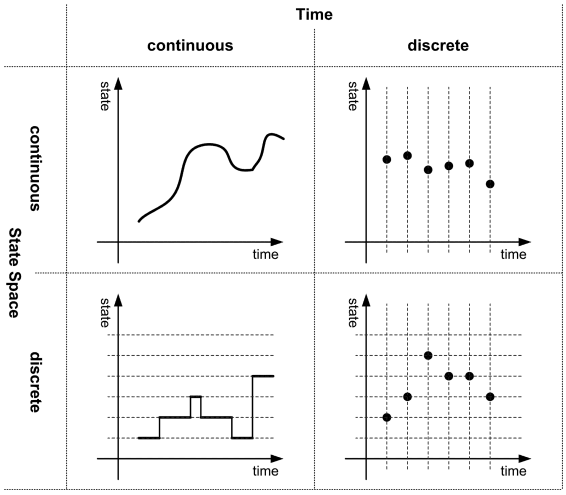

Figure 1: Discrete- vs. continuous-time and discrete vs. continuous state space models.

During discrete-time modeling, we only consider a discrete subset $T \subseteq \mathbb{R}$ for the points in time of interest. When we use continuous time, then the current macro-state may change anytime.

Modeling using a discrete macro-state space means that the set of macro-states S is a discrete set (usually a finite or countable subset of $\mathbb{R}^n$, where n is the number of dynamical variables). Models with discrete state space are often also referred to as *discrete-event* models [Jai91, p. 399]. For discussing discrete-event models, a vast amount of literature is available (see, e.g., [Jai91, BGDT06, CL08] and the references therein). When modeling a system with continuous macro-state space, the set S of macro-states is uncountable (usually an open subset of $\mathbb{R}^n$).

2.1 Discrete-Time Macro-Level Modeling

For discrete-time macro-level modeling, the time is discrete, while the state space may still be continuous or also discrete. Due to discrete time, each orbit contains only countable many states since from the (possibly uncountable) set of macro-states, only a countable subset can be chosen.

For simplification, the points in time where state changes may occur are here defined as a subset of natural numbers $T \subseteq \mathbb{N}$. The initial state of an orbit $(x_0, x_1, \ldots)$ is state x_0 at time $t = 0$. The behavior rule describes to which macro-state the system changes at each point in time of interest. For further simplification, we can assume that only the current state x_t is needed to compute the next state x_{t+1}, because older states x_s for $s < t$ can be coded into the current state by extending the state space. Again for simplification, we assume time homogeneity, i.e., the point in time t is not explicitly used in the behavior rules. In principle, any time-inhomogeneous model can be transferred to a time-homogeneous model by extending the macro-state space to $S \times T$ in order to have access to the current absolute point in time within the behavior rules.

Based on these simplifications, the behavior rule can be expressed as a *recurrence equation* (also known as *difference equation* [CL08]): $x_{t+1} = f_p(x_t)$ for a map $f_p : S \to S$, where $p = (p_1, p_2, \ldots, p_k)$ are the static system parameters. A deterministic map f_p resembles a deterministic system, i.e., the orbit is uniquely determined by the initial state and the system parameters. If f_p is a random variable, we handle a stochastic system. In this case, the orbit also depends on random events, i.e., each initial macro-state may lead to many different orbits.

An equilibrium of a deterministic discrete-time system is a solution of the equality $f_p(x) = x$. For the analysis which equilibria are stable, it might be difficult to compute the complete orbit $(x_t)_{t \in T}$, so other methods have been investigated in literature for the stability analysis [Boc04]. In the Euclidean space $S \subseteq \mathbb{R}^n$, the complex eigenvalues of the Jordan matrix $Df_p(x^*)$ for an equilibrium x^* may be used for this purpose: First, we have to distinguish between hyperbolic equilibria ($|\lambda| \neq 1$ for all complex eigenvalues λ of $Df_p(x^*)$) and non-hyperbolic equilibria ($|\lambda| = 1$ for some complex eigenvalue λ of $Df_p(x^*)$). A hyperbolic equilibrium x^* is asymptotically stable iff $|\lambda| < 1$ for each complex eigenvalue. If a hyperbolic equilibrium has $|\lambda| > 1$ for some eigenvalue λ, then the equilibrium is unstable.

2.1.1 Discrete-Time Macro-Level Modeling with Discrete State Space

Behavior rules of deterministic discrete-time macro-level models with discrete state space can be described by recurrence equations as described at the beginning of Section 2.1.

For building stochastic macro-level models with discrete state space, stochastic processes with discrete state space (also known as *chains*) can be used. Frequently employed discrete-time representatives of such chains are *discrete-time Markov chains* (DTMCs, cf. [BGDT06, p. 53]). Stable equilibria of (discrete-time) Markov chains are called *absorbing states* (cf. [BGDT06, p. 62]). Absorbing states can directly be identified by observing a diagonal element of value 1 within the DTMC's transition probability matrix $\mathbf{P}$. Moreover, attractors of the system can be considered as the highest value within the DTMC's steady-state probability vector (cf. [Dre07, p. 28] or [Hey03]). If a unique steady-state probability vector $\boldsymbol{\pi}$ exists, it can be determined by solving the system of equations $\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{P}, \sum_{i \in S} \pi_i = 1$. We refer the interested reader to [BGDT06, Chapter 2] for more details on Markov chains and their analysis.

2.1.2 Discrete-Time Macro-Level Modeling with Continuous State Space

Behavior rules of deterministic discrete-time macro-level models with discrete state space can also be described by recurrence equations as described in Section 2.1. An example for such a model is the *discrete logistic model* (cf. [Boc04, p. 128]).

For building stochastic models with continuous state spaces, stochastic processes can be used which can be seen as a generalization of (Markov) chains since the former do not rely on a discrete state space (see, e.g., [Tri01, BGDT06, CL08] for details on stochastic processes).

2.2 Continuous-Time Macro-Level Modeling

For continuous-time macro-level modeling, the time parameter space is considered continuous. Hence, a state change may occur anytime: $T \subseteq \mathbb{R}_0^+$, where x_0 is the initial state at time $t = 0$.

Here, an orbit of the system can be seen as the trajectory $(x_t)_{t\in T}$ of the macro-state during the time. The behavior rule describes the state change at each point in time. Like in the case of discrete macro-level modeling, we assume that older states are not needed to describe the behavior of the system and also the time is not explicitly needed in the rule. However, we cannot derive a recurrence equation here since the time is not discrete. For discussing the system's equilibria, we now need to differentiate between models with continuous and models with discrete state space.

2.2.1 Continuous-Time Macro-Level Modeling with Continuous State Space

For continuous-time macro-level models with continuous state space, a *differential equation* for the rules can be derived: $\dot{x} = f_p(x)$. An equilibrium is then characterized by the equality $f_p(x^*) = 0$. For stability analysis of equilibria [Boc04], the eigenvalues of the Jacobian matrix may again be used, like in the discrete-time case (Section 2.1): We similarly distinguish between hyperbolic equilibria (real part of each complex eigenvalue is non-zero: $Re(\lambda) \neq 0$) and non-hyperbolic equilibria (real part of some complex eigenvalue is zero: $Re(\lambda) = 0$). A hyperbolic equilibrium x^* is asymptotically stable iff $Re(\lambda) < 0$ for each complex eigenvalue. If a hyperbolic equilibrium has $Re(\lambda) > 0$ for some eigenvalue λ, the equilibrium is unstable. Examples for deterministic models are the *Lotka-Volterra model* [Boc04, p. 17] and the *logistic model* [Boc04, p. 6].

Similar to Section 2.1.2, stochastic processes can be employed to model continuous-time macro-level models with continuous state space.

2.2.2 Continuous-Time Macro-Level Modeling with Discrete State Space

Unfortunately, differential equations cannot be used for describing the behavior rules of continuous-time macro-level models with discrete state space due to the instantaneous jumps from one state one state to another (discrete events). However, in deterministic systems, the time instants when these jumps happen are pre-determined, and hence, the model can be mapped to a discrete-time macro-level model with discrete state space (see Section 2.1.1).

A common example for a stochastic continuous-time macro-level models with discrete state space are *continuous-time Markov chains* (CTMCs, cf. [BGDT06, p. 64]). Comparable to DTMCs (Section 2.1.1), stable equilibria (absorbing states) can directly be identified by observing a row of zeros within the CTMC's transition rate matrix $\mathbf{Q}$. Attractors of the continuous-time system can be considered as closely related to the CTMC's steady-state probability vector. If a unique steady-state probability vector $\boldsymbol{\pi}$ exists, it can be obtained by solving $\mathbf{0} = \boldsymbol{\pi}\mathbf{Q}, \sum_{i\in S} \pi_i = 1$.

2.3 Bifurcation

A change between qualitatively different behaviors by the variation of some system parameters p is called bifurcation [Boc04]. Such a change could be the *appearance of new equilibria*, the *elimination of existing equilibria*, the *change of the equilibria's stability*, the *appearance of new limit cycles*, or the *elimination of existing limit cycles*. Note that such a bifurcation can only appear for non-hyperbolic equilibria.

Let us first consider the different types of one-dimensional bifurcations, i.e., the system parameter p is a real number. In [Boc04], the following types of one-dimensional bifurcations

have been defined for discrete and continuous models: *saddle-node bifurcation*, *transcritical bifurcation*, and *pitchfork bifurcation*.

As an example, we are looking at a fish population and the effect of fishing. Let N be the size of the fish population, r be the birth rate, and K be the carrying capacity. Let C be the intrinsic catch rate. Using the logistic model for the birth of fishes, we get $\dot{N} = rN(1-\frac{N}{K}) - CN$. By introducing the dimensionless variables $n = \frac{N}{K}$, $\tau = rt$, $c = \frac{C}{r}$, we derive the new differential equation $\frac{dn}{d\tau} = n(1-n) - cn$. This system has two equilibria: $n_1^* = 0$ and $n_2^* = 1-c$. For $c = 1$, we have only one equilibrium at $n_1^* = n_2^* = 0$. The equilibrium $n_1^* = 0$ is asymptotically stable for $c > 1$, instable for $c < 1$, and non-hyperbolic for $c = 1$. The equilibrium $n_2^* = 1-c$ is asymptotically stable for $c < 1$, instable for $c > 1$, and non-hyperbolic for $c = 1$. Hence, the equilibria change their stability: The system has a transcritical bifurcation at $n^* = 0$ for $c^* = 1$.

In dimension two, there may also be other types of bifurcations. For example, the asymptotically stable equilibrium of a predator prey model (see [Boc04]) becomes a limit cycle, i.e., a closed orbit attracting other trajectories.

Bifurcation can also been observed in discrete systems. In addition to saddle-node, transcritical, and pitchfork bifurcations which also exist for discrete systems, there is another type of one-dimensional bifurcation: *Period-doubling bifurcation*. Here, an equilibrium is split into two different states which form a cycle of length two. Since each cycle of length d of the system $x_{t+1} = f_p(x_t)$ can be seen as an equilibrium of the system $y_{t+1} = f_p^d(y_t)$, this concept can also be used for doubling cycles of arbitrary length.

3 Micro-Level Modeling

By micro-level models the behavior of the objects in the system is described. For defining the micro-level model, the following items of the model have to be specified:

- Topology G: The topology describes which object is able to communicate with which other objects. If the topology is static, a graph $G = (V,E)$ (or multigraph) can be used for this purpose, where each node $v \in V$ of the graph represents an object of the system and each edge $e \in E$ represents a communication channel. The graph may be directed, if the communication channels are not symmetric. If the topology changes dynamically, we could either use a time dependent graph G_t for each point of time t or we could use a graph G to describe the boundaries of the changing topology, i.e., G contains all nodes and communication channels which may be existent at any point in time.

- Micro-state space S_v for each node $v \in V$: At each point in time, the node $v \in V$ has an internal state $s \in S_v$.

- Set $T \subseteq \mathbb{R}$ specifying the points in time the state of a node may change. State changes may occur discretely (e.g., $T = \mathbb{N}$) or continuously (e.g., $T = \mathbb{R}_0^+$). Time $t = 0$ usually refers to the system's initialization.

- Alphabet A for communication: At each point in time $t \in T$, each node can send data to his successor nodes. The alphabet A describes the possible values for this communication.

- Behavior rules $R = (R_v)_{v \in V}$ for the nodes: The change of the internal states of the nodes and the communication between the nodes must be described by rules.

Like macro-level modeling, we also have to decide during micro-level modeling which aspects of the model should be discrete or continuous. There are four main aspects where this decision has a large impact on the model design and behavior: time T, object set V, states of objects S_v, and interaction. If we use discrete interaction, each object is able to interact with only a finite or countable number of other objects. A continuous interaction can be seen as a force or an impression of one object to other objects, e.g., gravitation force of a planet.

3.1 Discrete-Time Micro-Level Modeling

Here, the behavior of each object $v \in V$ can be described by a deterministic or stochastic automaton a_v: At each point in time $t \in T$, the node v gets some local input from its predecessor nodes $v- := \{w \in V \mid (w,v) \in E\}$, looks at its current internal state, and then decides (deterministically or stochastically) on the state change and on the local output to the successor nodes $v+ := \{w \in V \mid (v,w) \in E\}$. A stochastic automaton $a_v = (A^{v-}, A^{v+}, S_v, P_v)$ for $v \in V$ consists of

- the local input values $A^{v-} = \{(x_w)_{w \in v-} \mid x_w \in A, w \in v-\}$,
- the local output values $A^{v+} = \{(x_w)_{w \in v+} \mid x_w \in A, w \in v+\}$,
- the set S_v of states, and
- a map $P_v : S_v \times A^{v-} \times S_v \times A^{v+} \to [0,1]$, such that $P(q,x,\cdot,\cdot) : S_v \times A^{v+} \to [0,1]$ is a probability distribution on $S_v \times A^{v+}$ for each $q \in S_v$ and $x \in A^{v-}$. The value $P(q,x,q',y)$ is the probability that the automaton moves from state $q \in S_v$ into the new state $q' \in S_v$ and gives the local output $y \in A^{v+}$ when it receives the local input $x \in A^{v-}$.

A deterministic automaton can be seen as a special case of such a stochastic automaton, where the probabilities are either 0 or 1. It is also possible to model the clocks of the automata as non-synchronous [HD09]: In this case, the set T is not given in advance, but a map $d_v : S_v \to \mathbb{R}^+$ describes the delay between two pulses of the clock when the automaton is in a given state $q \in S_v$, so each automaton has its own clock which depends on the current state.

In the global view on a micro-level model, the current configuration of the whole system can be seen: A global state s consists of all local states and all values on the communication channels.

3.2 Continuous-Time Micro-Level Modeling

A change of the nodes' states may now appear anytime. One possibility to specify the nodes' behavior in such a system is given in [HD08]: The nodes' behavior contains the state change and the local output of the nodes. A continuous state change can be described by differential equations. For this purpose, we assume that the state space S_v is a subset of a normed vector space for each node $v \in V$. Like the map f_p for macro-level modeling, the right hand side of the differential equation can be described by a family $f = (f_v)_{v \in V}$ of (deterministic or stochastic) maps $f_v : A^{v-} \times S_v \to S_v$, where f_v is called *change map of the object* v. For the local output, we

can also use a family $\lambda = (\lambda_v)_{v \in V}$ of maps $\lambda_v : S_v \times v+ \to A$, where λ_v is called *output map of the object v*. If also non-continuous changes of states should be modeled, we need another family $h = (h_v)_{v \in V}$ of maps $h_v : A^{v-} \times S_v \to S_v$, where h_v is called *hop map of the object v*.

A family $(s_v)_{v \in V}$ of maps $s_v : \mathbb{R}_0^+ \to S_v$ is called *behavior* of the system with respect to an initialization $I \in \prod_{v \in V} S_v$, if for all $v \in V$

(B1) $s_v(0) = I_v$,

(B2) s_v is left-continuous,

(B3) $\{t \in \mathbb{R}_0^+ \mid s_v \text{ is not differentiable in } t\}$ is a discrete set,

(B4) for each $t \in \mathbb{R}_0^+$, for which $s_v(t)$ is differentiable, we have $\dot{s}_v(t) = f_v((\lambda_w(s_w(t), v))_{w \in v-}, s_v(t))$, and

(B5) $\lim_{r \searrow t} s_v(r) = s_v(t) + h_v((\lambda_w(s_w(t), v))_{w \in v-}, s_v(t))$ for $t \in \mathbb{R}_0^+$.

Therefore, the behavior s_v of the node v is just the local orbit of the internal state of v which results from the solution of the differential equation (B4) and the hops given by (B5).

3.3 Network Properties

For very complex systems, the topology is usually not known in every detail. In such a case, the graph G is not specified deterministically, but it is considered as a random graph, where only some static system parameters (e.g., number of nodes) and probability distributions (e.g., the probability that an edge exists between two randomly chosen nodes) are specified. For such a model, it is not easy to find the appropriate probability distributions that are needed to specify the model, since the distributions strongly depend on the network properties of the corresponding system in the real world. To be able to find a good model, many network properties of the real system can be analyzed:

- The mean value L for the path length between two randomly chosen nodes,
- the mean size and mean diameter of maximal connection component,
- the clustering coefficient C, i.e., the conditional probability that two randomly selected vertices are connected given that they are both connected to a common vertex,
- and the average node degree.

After calculating these network properties, it turns out that for most applications a simple uniformly distributed random graph is not a good system model, since such a random graph has completely different properties. One kind of graphs that appear very often in practice are small-world networks: They have short characteristic path lengths and high clustering coefficients. To find an adequate model for such a network, some algorithms have been proposed in literature. The model of Watts and Strogatz [WS98] starts with a lattice structure and rewires each edge with a constant probability p, where the new target node is chosen randomly. It can be shown

that for small values of p, the Watts-Strogatz model has a clustering coefficient of the same order of magnitude as the lattice, while its characteristic path length is of the order of magnitude of a uniform random graph. Other algorithms for small-world networks can be found in [Boc04].

Since the small world networks are characterized by mean path length and clustering coefficient, there are still some important properties where the model differs to the corresponding system in the real world. One of these properties is the node degree distribution. Many systems in the real world are scale-free, i.e., the node degree distribution follows the power law $P(d(v) = k) \sim k^{-\gamma}$ for some constant γ, where $d(v)$ is the degree of node $v \in V$. Also for this kind of networks, some algorithms have been proposed in literature. Usually, these algorithms use the two principles *expansion* (the size of real networks are not constant, but it usually continuously increases during the time) and *preferential attachment* (in growing real networks, a new vertex has a higher probability of being connected to a vertex that already has a large degree). Barabási and Albert [AB02] have proposed such an algorithm to construct a scale-free network. The algorithm starts with a graph without any edges. In each step, a new vertex and m new edges connecting this vertex to m existing vertices are added to the current graph using preferential attachment for the probabilities of the target nodes. This leads to a scale-free network with the power law constant $\gamma \approx 2.9$. Other algorithms for scale-free networks can be found in [Boc04].

4 From Micro-Level to Macro-Level: Quantitative Measures

Since a macro-level model considers only the variables of interest of a system, it is also possible to first specify the micro-level model and then define the variables of interest as numeric attributes in the micro-level model. For this purpose, each global state in the micro-level model must be mapped into the set of real numbers. Such a map is called *quantitative measure*. This concept can be used to analyze the system with respect to properties of interest. In [HDB08, HD09, AWD09b, AWD09a], this has been done for the properties *autonomy*, *emergence*, *adaptivity*, *resilience*, *homogeneity*, *target orientation*, and *state classification*. One important tool for such quantitative measures is the statistical entropy $H(X) = -\sum_{w \in W} P(X = w) \log_2 P(X = w)$ of a random variable X that measures how many bits are needed to encode the outcome of the random variable in an optimal way. It can be used to measure different properties of the system:

- How much information is contained in the system at time $t \in T$?
- How much control information is needed at time $t \in T$ to keep the system running?
- How much output does the system produce at time $t \in T$?

For example, to measure the emergence of global patterns in the system, the dependencies between the communications of different nodes can be analyzed by comparing the information of the whole system with the sum of the information of each edge [HDB08]: The ratio $\frac{H(\text{information on all edges})}{\sum_{k \in E} H(\text{information on } k)}$ indicates how many dependencies are in the communications of the system. Such quantitative measures help to analyze existing systems and to design new systems.

5 From Macro-Level to Micro-Level: Design Approaches

When we design a new system, we first have a target function in mind that should be fulfilled by the system. The description of this target function may be easily achieved on the macro-level. However, it is a non-trivial task to transform the macro-level target function into the corresponding micro-level rules in general (see also Design Paradigm #1 described in [PB05]).

In [ED08], three different design approaches are sketched: the *bio-inspired design*, the *trial-and-error approach*, and the *design by learning from an omniscient entity*.

The *bio-inspired design* approach tries to exploit the mechanisms developed by nature with the help of long-term evolution. The approach can be direct (top-down) or indirect (bottom up). The direct approach aims at searching for natural blueprints of mechanisms that achieve solving a problem that is similar to the technical problem under study. The indirect approach first studies a natural phenomenon and then searches for an application of this phenomenon in the technical world. However, the bio-inspired design approach is infeasible when the technical implementation of the natural solution is too intricate or the solution can be technically implemented more efficiently by exploiting approaches that are not available in biological systems. An example for bio-inspired design is the class of Ant Routing Algorithms (see, e.g., [GSB02]).

The *trial-and-error* approach requires a first implementation of the target system in form of a testbed (or simulation). During the design process, extensive evaluation of the testbed's ability to provide the target functionality is conducted. In this testing process, the system parameters are (purposefully) varied to approach a suitable (or even close to optimal) setting. However, the search space for such settings usually is very large and often the search ends in a local optimum. Hence, in general, following this approach is a very time- and resource-consuming tasks.

The approach by *learning from an omniscient entity* in introduced in [AWD08]. A generic method is proposed that disburdens the derivation of (close to optimal) micro-level rules by providing a subset of a testbed's system entities with extended information on global system and environmental properties. The extended information available to these "omniscient" entities (referred to as *Laplace's Daemons* (LDs)) simplifies the design of (close to optimal) entities but will not be available to the entities during the operation of the real system. The micro-level behavior, i.e., the series of local inputs and outputs, of these LDs is then analyzed by using an algorithm for time series analysis (CSSR algorithm; see [SS04]). The method results in a Markov chain description of each investigated LD which can then be used as a blueprint for the micro-level behavior of the entities in the real system.

6 Conclusion

Mathematical models are useful for both the analysis and design of systems. In this paper, we have given an overview on different modeling methods for self-organizing systems. Micro-level modeling can be used to describe the behavior of each object in the system and the communication between the objects. It's advantage is that more details of the real system can be integrated into the model. But this leads to a very large global state space for the model. Macro-level modeling abstracts from the individual entities of the system and models only the variables of interest. Hence, the large state space of the micro-level is strongly reduced by considering equiv-

alence classes. For both approaches, some model properties can be specified in different ways: discrete vs. continuous time, discrete vs. continuous space, and deterministic vs. stochastic behavior. The choice of the right model properties depend on the application and on the aspired analysis. Each of these models may be useful for a better understanding of the corresponding real-world system and also help in the design process. Note that the choice of the model may be done independently of the properties of the real system. For example, many continuous-state population models have been developed in literature (see, e.g., [Boc04]) although the size of the population in the real system are natural numbers. Also in IT, where usually everything is considered being discrete, continuous-state continuous-time models can be used for system analysis (see e.g. [HD08]). A case study may be interesting where some of the models are applied to the same problem to be able to compare the different approaches and to get guidelines which modeling approach fits certain applications best. Due to lack of space such a case study is not be included in this paper, but some of the approaches discussed in this paper have been applied to a synchronization algorithm in wireless sensor networks in [HD09], [HDB08], [HD08], and [AWD09a], so a comparison between the models and their advantages are left as future work.

Bibliography

[AB02] R. Albert, A.-L. Barabasi. Statistical Mechanics of Complex Networks. *Review of Modern Physics* 74:47–97, 2002.

[AWD08] C. Auer, P. Wüchner, H. De Meer. A Method to Derive Local Interaction Strategies for Improving Cooperation in Self-Organizing Systems. In Hummel and Sterbenz (eds.), *3rd Int'l Workshop on Self-Organizing Systems (IWSOS 2008)*. Lecture Notes in Computer Science (LNCS) 5343, pp. 170–181. Springer Verlag, Vienna, Austria, December 2008.

[AWD09a] C. Auer, P. Wüchner, H. De Meer. The Degree of Global-State Awareness in Self-Organizing Systems. In Thrasyvoulos and Hummel (eds.), *4th Int'l Workshop on Self-Organizing Systems (IWSOS 2009)*. Lecture Notes in Computer Science (LNCS) 5918, pp. 125–136. Springer Verlag, Zurich, Switzerland, December 2009.

[AWD09b] C. Auer, P. Wüchner, H. De Meer. Target-Oriented Self-Structuring in Classifying Cellular Automata. In Oliviera and Kari (eds.), *Proc. of 15th International Workshop on Cellular Automata and Discrete Complex Systems (Automata2009)*. Pp. 260–271. Universidade Presbiteriana Mackenzie, Sao Paulo, SP, Brazil, 2009.

[BGDT06] G. Bolch, S. Greiner, H. De Meer, K. Trivedi. *Queueing Networks and Markov Chains*. John Wiley & Sons, New York, 2nd edition, 2006.

[Boc04] N. Boccara. *Modelling Complex Systems*. Springer, 2004.

[CL08] C. G. Cassandras, S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 2nd edition, 2008.

[DK05] H. De Meer, C. Koppen. Characterization of Self-Organization. In Steinmetz and Wehrle (eds.), *Peer-to-Peer Systems and Applications*. Lecture Notes in Computer Science (LNCS) 3485, chapter 15. Characterization of Self-Organization, pp. 227–246. Springer-Verlag, 2005.

[Dre07] F. Dressler. *Self-Organization in Sensor and Actor Networks*. John Wiley & Sons, 2007.

[ED08] W. Elmenreich, H. De Meer. Self-Organizing Networked Systems for Technical Applications: A Discussion on Open Issues. In Hummel and Sterbenz (eds.), *Proc. of the 3rd International Workshop on Self-Organizing Systems (IWSOS '08), Vienna, Austria*. LNCS 5343, pp. 1–9. Springer Verlag, Vienna, Austria, December 2008.

[GSB02] M. Güneş, U. Sorges, I. Bouazizi. ARA – The Ant-Colony Based Routing Algorithm for MANETs. In *InternationalWorkshop on Ad Hoc Networking (IWAHN 2002)*. Vancouver, British Columbia, Canada, August 2002.

[HD08] R. Holzer, H. De Meer. On Modeling of Self-organizing Systems. In *Autonomics 2008*. 2008.

[HD09] R. Holzer, H. De Meer. Quantitative Modeling of Self-Organizing Properties. In *IWSOS 2009*. LNCS. Springer, 2009.

[HDB08] R. Holzer, H. De Meer, C. Bettstetter. On Autonomy and Emergence in Self-Organizing Systems. In Hummel and Sterbenz (eds.), *IWSOS 2008*. LNCS 5343. Springer, 2008.

[Hey03] F. P. Heylighen. The Science of Selforganization and Adaptivity. In Kiel (ed.), *Knowledge Management, Organizational Intelligence and Learning, and Complexity*. The Encyclopedia of Life Support Systems. EOLSS Publishers, 2003.

[Jai91] R. Jain. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. John Wiley & Sons, 1991.

[PB05] C. Prehofer, C. Bettstetter. Self-organization in communication networks: principles and design paradigms. *Communications Magazine, IEEE* 43(7):78–85, 2005.

[SS04] C. R. Shalizi, K. L. Shalizi. Blind construction of optimal nonlinear recursive predictors for discrete sequences. In *AUAI '04: Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*. Pp. 504–511. AUAI Press, Arlington, Virginia, United States, 2004.

[Tri01] K. S. Trivedi. *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*. John Wiley & Sons, 2 edition, nov 2001.

[WS98] D. J. Watts, S. H. Strogatz. Collective Dynamics of 'Small-World' networks. *Nature* 393:440–442, 1998.

[ZD97] K. Zhou, J. Doyle. *Essentials of Robust Control*. Prentice Hall, 1997.