# Modeling and Application of Self-organizing Systems
# Tutorial paper

Richard Holzer and Hermann de Meer
*University of Passau*
*Chair for Computer Networks and*
*Computer Communications*
*Innstrasse 43*
*94032 Passau, Germany*
*Email: holzer@uni-passau.de*

***Abstract*—This tutorial gives an overview about mathematical modeling methods for complex and self-organizing systems. Modeling can be used for the analysis and optimization of existing systems and for the design and engineering of new systems. In this tutorial we classify modeling methods into macro-level modeling and micro-level modeling. By using a micro-level model, the behaviors of all entities of the system and the interactions between these entities have to be specified. The state space of such a model is the Cartesian product of the state spaces of each entity. For a macro level, many micro-level states are aggregated into a single macro-level state. The macro level model describes only the behavior of the variables of interest. Another classification for modeling methods is the time space: The advance of time can either be modeled discrete or continuous. This tutorial contains short introductions to some modeling methods (e.g. Markov chains, cellular automata, recurrence equations, differential equations, . . . ) and a discussion about their possibilities for analysis, optimization, design and engineering of self-organizing systems. The applicability of the modeling methods are demonstrated in some use cases.**

***Keywords*-Self-Organization, Modeling, Systems**

## I. Introduction

This tutorial gives an overview about mathematical modeling methods for complex and self-organizing systems. Observations and measurements in the real system might be impossible or infeasible because:

- It does not exist (e.g. the system is in the design process),
- the Hardware of the system is not available to the modeler,
- it is too expensive to experiment with (e.g. downtimes for doing measurements result in lack of profit or an extra system for testing is highly priced),
- it is too dangerous to experiment with directly (e.g. high-voltage systems, military systems).

For the analysis, evaluation and optimization of a system a mathematical model is useful. With the model, new policies, decision rules, information flows, etc. can be explored without disrupting ongoing operations of the real system. New hardware architectures, scheduling algorithms, routing protocols, reconfiguration strategies, etc., can be tested without binding resources for their acquisition or implementation. The evaluation of systems under a wide variety of workload and network types or protocols can be carried out without excessive costs. Models can be used for suggesting improvements to the real system under investigation based on knowledge already gained during modeling. Models can be used to gain insight into which system parameters are most important and how these parameters interact. But there is no perfect model apart from the real system. The modeling process is a complex art. The artists (modelers) need creativity and experience to fulfill this task.

In this tutorial we distinguish between micro-level modeling and macro-level modeling. While in micro-level modeling the behavior of each entity and their interactions have to be specified, macro-level modeling uses the concept of aggregation to reduce the global state space. In macro-level modeling only the behavior of the variables of interest have to be specified. Section II presents methods for macro-level modeling and Section III covers micro-level modeling methods. Section IV describes evaluation methods for micro-level models based on quantitative measures. In section V we apply some of these evaluation methods in an wireless synchronization example. Section VI concludes this tutorial paper. The details of the topics covered by this tutorial can be found in [1], [2], [3], [4] and [5].

## II. Macro-level modeling

In macro-level modeling we abstract from the individual entities of the system and look only on the variables of interest. We specify some rules for the change of these variables during the time. Then we can analyze the behavior of these variables during the whole run of the system. We distinguish between continuous-time systems and discrete-time systems.

A dynamical system consists of

- a state space S,

R. Holzer, H. de Meer, Modeling and Application of Self-organizing Systems - Tutorial paper. In: Proc. of the Fifth IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO), IEEE Computer Society Press, 2011, to appear.

- a set $T \subseteq \mathbb{R}$ of points $t \in T$ in time at which a macro-state change of the system can be observed,
- an evolution law, which describes the change of the state during the time.

The state space contains all possible states of the system. If more than one value is needed to describe the state, then it can be of higher dimension, e.g. each state $s = (s_1, \dots, s_n) \in S$ is a tuple.

## A. Continuous time macro-level modeling

In a time-continuous dynamic system, the evolution law is usually described by a system of differential equations. We can assume, that each differential equation has order 1, because a higher order differential equation $y^{(n)} = f(t, y, \dot{y}, \dots, y^{(n-1)})$ can always be transformed into a system of differential equations of order 1 by introducing new variables $z_0 = y, z_1 = \dot{y}, \dots, z_{n-1} = y^{(n-1)}$. As an example we consider a damped pendulum (see Fig. 1). Let

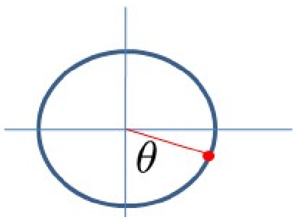

Figure 1. Damped pendulum

$g$ be the acceleration of gravity and $l$ be the length of the pendulum. $\theta$ is the angle between the pendulum and the vertical axis. Let $a$ be the friction coefficient. Then the rule for the change of the system variable $\theta$ can be described by a differential equation of second order:

$$\ddot{\theta} + \frac{g}{l}\sin\theta + 2a\dot{\theta} = 0$$

We introduce new variables $x_0 = \theta,\ x_1 = \dot{\theta}$ and get the system of differential equations of order 1:

$$\begin{aligned}\dot{x}_0 &= x_1\\ \dot{x}_1 &= -\frac{g}{l}\sin x_0 - 2ax_1\end{aligned}$$

A differential equation of the form

$$\dot{y} = Y(y, t)$$

in which the time variable also occurs on the right side can be transformed into differential equations of the form

$$\dot{x} = X(x)$$

by introducing $x_1 = y,\ x_2 = t$, which leads to the differential equation system

$$\begin{aligned}\dot{x}_1 &= Y(x_1, x_2)\\ \dot{x}_2 &= 1\end{aligned}$$

Here, the time is stored in the state space.

In the following we assume that the dynamic system has the form $\dot{x} = X(x)$ for a vector field $X : \mathbb{R}^n \to \mathbb{R}^n$ and that the solution $\phi_{x_0}$ of this system is uniquely determined by the initial condition $x(0) = x_0$. The solution can be drawn as an orbit in $\mathbb{R}^n$, i.e. we have the trajectory $\{\phi_{x_0}(t) \mid t \geq 0\}$.

A point $x^* \in \mathbb{R}^n$ is an equilibrium point of the system $\dot{x} = X(x)$, if $X(x^*) = 0$. Each equilibrium point is a fixed point of the flow: $\phi_{x^*}(t) = x^*$ for all $t \geq 0$. The orbit of a fixed point is the fixed point itself.

Let $x^*$ be an equilibrium. $x^*$ is stable, if each trajectory starting near $x^*$ will stay near $x^*$: For all $\varepsilon > 0$ there exists $\delta > 0$ such that for all $x$ with $||x - x^*|| < \delta$ we get $||\phi_x(t) - x^*|| < \varepsilon$ for all $t > 0$. Otherwise the equilibrium is unstable. $x^*$ is asymptotically stable, if it is stable and each trajectory starting near $x^*$ will converge to $x^*$: There exists $\delta > 0$ such that for all $x$ with $||x - x^*|| < \delta$ we get $\lim\limits_{t\to\infty} \phi_x(t) = x^*$. An equilibrium $x^*$ is neutrally stable, if it is stable but not asymptotically stable.

How can we analyze the behavior of a dynamic system? One possibility is to solve the differential equation. But this is usually very difficult and in many cases it is impossible. Often it is not necessary to know the solution for the analysis. In the following we will see, how the behavior can be derived directly from the differential equation. For a dynamical system $\dot{x} = X(x)$ we consider the Taylor approximation of first order:

$$T_1(x) = X(a) + DX(a) \cdot (x - a)$$

where

$$DX = \begin{pmatrix} \frac{\partial X_1}{\partial x_1} & \dots & \frac{\partial X_1}{\partial x_n} \\ \dots & \dots & \dots \\ \frac{\partial X_n}{\partial x_1} & \dots & \frac{\partial X_n}{\partial x_n} \end{pmatrix}$$

is the Jacobian Matrix. The Eigenvalues of the Jacobian matrix provide information about the stability of equilibria. The nonlinear differential equation $\dot{x} = X(x)$ can been transformed into a linear differential equation by Taylor approximation and substitution $y := x - x^*$, where $x^*$ is an equilibrium of the system: $\dot{y} = DX(x^*) \cdot y$. The map $x \mapsto DX(x^*) \cdot x$ is called linear part of $X$. The equilibrium $x^*$ is called hyperbolic, if all (complex) eigenvalues of $DX(x^*)$ have a nonzero real part. For a hyperbolic equilibrium, the transformation to the linear differential equation does not change the stability:

- $x^*$ is stable for $\dot{x} = X(x)$ iff 0 is stable for $\dot{y} = DX(x^*) \cdot y$
- $x^*$ is asymptotically stable for $\dot{x} = X(x)$ iff 0 is asymptotically stable for $\dot{y} = DX(x^*) \cdot y$
- The hyperbolic equilibrium $x^*$ is never neutrally stable.

Since the stability of a linear differential equation can be easily analyzed, these properties yield the characterization of the stability of hyperbolic equilibria:

- A hyperbolic equilibrium $x^*$ is (asymptotically) stable iff all eigenvalues of $DX(x^*)$ have a negative real part.
- A hyperbolic equilibrium $x^*$ is unstable iff at least one eigenvalue of $DX(x^*)$ has a positive real part.

Let us now consider some other examples. A population model describes the changes of the size $N$ of the population of a species in dependency of the time $t$. We abstract from the behavior of each individual and assume, that the growth of the population only depends on the number of the individuals and some constant system parameters. One of the simplest population model is the exponential growth: $\dot{N} = r \cdot N$. The system parameter $r$ is the rate of increase for the population size $N$. The solution of this differential equation is $N(t) = ae^{rt}$, where $a = N(0)$ is the initial value. Now we can restrict the growth by a capacity $K$, which defines the maximal number of individuals in the system such that $N$ can not exceed $K$. This leads to the logistic model: $\dot{N} = rN(1 - \frac{N}{K})$. The logistic model can be simplified by normalizing the variables: $\tau = rt$ and $n = \frac{N}{K}$. We get the dimensionless logistic model $\frac{dn}{d\tau} = n(1-n)$. This equation does not depend on the properties of the special system anymore. The solution of this differential equation is $n(\tau) = \frac{1}{1+ae^{-\tau}}$, where the constant $a$ is determined by the initial value: $a = \frac{1-n(0)}{n(0)}$. The population $n(\tau)$ in dependency of the time is shown in Figure 2.

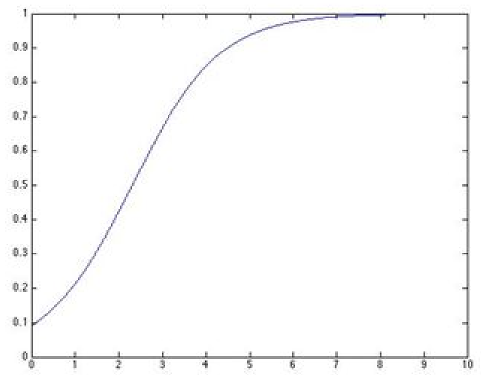

Figure 2. Dimensionless logistic model: $n$ in dependency of the time $\tau$

Now we consider two species: Predators and preys. We are interested in the change of the number of preys $B$ during the time. We use again the logistic model for the growth of $B$. The efficiency of predation $p(B)$ depends on the number of preys. At high prey density, predation usually saturates, so $p(B)$ should approach an upper limit $a > 0$, when $B$ becomes very large. At low prey density, predation is less effective. If a prey becomes less common, the predators seek food elsewhere. So $p(B)$ should tend to zero faster than $B$, when $B$ tends to zero. We use $p(B) = \frac{aB^2}{b^2+B^2}$ and get the dynamical system

$$\frac{dB}{dt} = rB(1 - \frac{B}{K}) - p(B) = rB(1 - \frac{B}{K}) - \frac{aB^2}{b^2 + B^2}$$

where $a$ describes the upper limit of the predation and $b$ describes the critical density of the prey. The function $p(B)$ tends to the value $a$ asymptotically for $B \to \infty$ and $p(B)$ tends to 0 like $B^2$. The differential equation can be simplified by introducing dimensionless variables:

$$\tau = \frac{t}{t_0}, \quad x = \frac{B}{B_0}$$

We have different possibilities to choose $t_0$ and $B_0$ in terms of $r, K, a, b$. If we want to analyze the influence of the predators on the preys, we can normalize the first term $rB(1-\frac{B}{K})$ and use the second term $\frac{aB^2}{b^2+B^2}$ for the analysis. If we want to analyze the influence of the environment capacity on the preys, then we can normalize the second term $\frac{aB^2}{b^2+B^2}$ and use the first term $rB(1 - \frac{B}{K})$ for the analysis.

In the first case we use $t_0 = \frac{1}{r}$ and $B_0 = K$ to get the dynamical system $\frac{dx}{d\tau} = x(1-x) - \frac{\alpha x^2}{\beta^2+x^2}$, where $\alpha$ is a scaled upper limit of the predation and $\beta$ is a scaled critical density of preys. The term for the influence of the environment capacity has been normalized. In the second case we use $t_0 = \frac{b}{a}$ and $B_0 = b$ to get the dynamical system $\frac{dx}{d\tau} = sx(1 - \frac{x}{k}) - \frac{x^2}{1+x^2}$ where $s$ is a scaled increase rate of preys and $k$ is a scaled capacity of preys. Here, the term for the predation has been normalized. Note that this system is self-organizing: There is no central instance controlling the birth, death or predation and no individuals needs to know the global state.

Now we consider the population model of Lotka Volterra, which also models two different types of species (preys and predators). Let $H$ be the number of preys and $P$ be the number of predators. A birth rate $b$ for the prays describes the rate of increase of the variable $H$. A death rate $d$ for the predators describes the rate of decrease of the variable $P$. Predation leads to an increase of the population of predators and a decrease of the population of the preys. Let $s$ be the parameter describing the efficiency of predation. Let $e$ be the parameter describing the increase of predators after successful predation. Then we can describe the evolution law with the following differential equation system:

$$\dot{H} = bH - sHP$$
$$\dot{P} = -dP + esHP$$

We can simplify this model by introducing new variables:

$$\begin{array}{rclcrcl} h & = & \frac{Hes}{d} & \quad & p & = & \frac{Ps}{b} \\ \tau & = & \sqrt{bd}t & \quad & \rho & = & \sqrt{\frac{b}{d}} \end{array}$$

which leads to

$$\frac{dh}{d\tau} = \rho h(1-p)$$
$$\frac{dp}{d\tau} = -\frac{1}{\rho} p\rho h(1-h)$$

This equation system depend only on one parameter: $\rho$ measures the ratio of the birth of preys to the death of predators. There are two equilibria: $(h_1^*, p_1^*) = (1,1)$ is neutrally stable and $(h_2^*, p_2^*) = (0,0)$ is instable.

Let us consider again the example of the damped pendulum.

$$\dot{x}_0 = x_1$$
$$\dot{x}_1 = -\frac{g}{l}\sin x_0 - 2ax_1$$

The equilibria are $(n\pi, 0)$ for $n \in \mathbb{Z}$. The Jacobian matrix is

$$DX(x_0,x_1) = \begin{pmatrix} 0 & 1 \\ -\frac{g}{l}\cos x_0 & -2a \end{pmatrix}$$

For even $n$ the Jacobian matrix $DX(n\pi, 0)$ has the Eigenvalues $\lambda = -a \pm \sqrt{a^2 - \frac{g}{l}}$, so the real part of the Eigenvalues are negative and the equilibrium $(n\pi, 0)$ is asymptotically stable. For odd $n$ we get the Eigenvalues $\lambda = -a \pm \sqrt{a^2 + \frac{g}{l}}$, so one positive real Eigenvalue and one negative real Eigenvalue. In this case the equilibrium $(n\pi, 0)$ is unstable.

Another analysis method for the stability of equilibria is based on Lyapunov functions: Let $x^*$ be an equilibrium. Let $V$ be a real function defined on a neighborhood $U$ of $x^*$ with

- $V(x) > V(x^*)$ for all $x \in U$
- $\frac{d}{dt}V(\phi_t(x))|_{t=0} < 0$ for all $x \in U \setminus \{x^*\}$

Then $V$ is called strong Lyapunov function for the flow. If the second condition is only $\frac{d}{dt}V(\phi_t(x))|_{t=0} \leq 0$ then $V$ is called weak Lyapunov function for the flow. Then the following properties can be shown:

- If there exists a weak Lyapunov function, then $x^*$ is stable.
- If there exists a strong Lyapunov function, then $x^*$ is asymptotically stable.

As an example we consider the Van de Pol oscillator, which is a harmonic oscillator with a nonlinear friction term: $\ddot{x} + \lambda(x^2-1)\dot{x} + x = 0$. This equation is transformed into a system of equations first order:

$$\dot{x}_0 = x_1$$
$$\dot{x}_1 = -x_0 - \lambda(x_0^2 - 1)x_1$$

The equilibrium is $(0,0)$. The function $V(x_0,x_1) := \frac{1}{2}(x_0^2 + x_1^2)$ satisfies $\dot{V}(x_0,x_1) = x_0\dot{x}_0 + x_1\dot{x}_1 = -\lambda(x_0^2-1)x_1^2$. In the neighborhood of the equilibrium $(0,0)$ we have $(x_0^2 - 1)x_1^2 < 0$, so $\dot{V}(x) < 0$ iff $\lambda < 0$. The equilibrium $(0,0)$ is asymptotically stable for $\lambda < 0$.

The advantages of this method are:

- It can be used for arbitrary differential equations.
- The solution of the differential equation need not be known.

The disadvantage is that we have to guess the Lyapunov function. But in many physical systems the energy of the system is a Lyapunov function. This was also the case in the example of the van de Pol oscillator.

As a summary, the following algorithm describes how a complex system can be modeled and analyzed:

- Description of the system by local interaction rules
  - Global dynamic parameters can be part of the local rules in form of a density to describe probabilities or rates of local events.
  - Global constants can be part of the local rules in form of system parameters.
  - Definition of global variables, which are interesting for analysis.
- From the local rules, the definition of the differential equations for these global variables can be derived.
- Calculate the equilibria
- Calculate the Jakobi matrix
- If the differential equality is linear, the stability can be derived from the eigenvalues of the matrix.
- If the differential equality is not linear:
  - If the equilibrium is hyperbolic, then the stability can be derived from the eigenvalues of the matrix.
  - Otherwise search for a Lyapunov function

## *B. Discrete time macro-level modeling*

It is also possible to use a discrete time variable for building the model. The set $T$ of all points in time is a discrete set, usually $T = \mathbb{N}_0 = \{0,1,2,\ldots\}$. The rule for the change of the system variables can not be given by a differential equation, but by a recurrence equation: $x_{t+1} = f(x_t)$, where $f : J \to J$ for a set $J \subseteq \mathbb{R}^d$. The orbit of a starting point $x \in J$ is $\{f^n(x) \mid n \in \mathbb{N}_0\}$. A point $x^*$ is an equilibrium point (fixed point) if $f(x^*) = x^*$. The orbit of the equilibrium is $\{x^*\}$. $x$ is a periodic point if $f^k(x) = x$ for some $k > 0$ The period length is the size of orbit $\{x, f(x), \ldots, f^{k-1}(x)\}$.

Let $x^*$ be an equilibrium. $x^*$ is stable, if each trajectory starting near $x^*$ will stay near $x^*$: For all $\varepsilon > 0$ there exists $\delta > 0$ such that for all $x$ with $||x - x^*|| < \delta$ we get $||f^t(x) - x^*|| < \varepsilon$ for all $t > 0$. Otherwise the equilibrium is unstable. $x^*$ is asymptotically stable, if it is stable and each trajectory starting near $x^*$ will converge to $x^*$: There exists $\delta > 0$ such that for all $x$ with $||x - x^*|| < \delta$ we get $\lim_{t\to\infty} f^t(x) = x^*$

Let $J$ be a closed set and $f$ be a contracting map, i.e. $||f(x) - f(y)|| \leq c||x-y||$ for all $x, y \in J$ for some constant $c < 1$ Then the following properties hold:

- There exists a unique equilibrium $x^*$.
- $x^*$ is asymptotically stable.

- For every starting point $x$ the sequence $(f^t(x))_{t\geq 0}$ converges exponentially to the equilibrium.

In the following we assume that $f : J \to J$ is differentiable. Like in the continuous case we consider the linear part $x \mapsto Df(x)$ of the map $f$. An equilibrium $x^*$ is hyperbolic, if the (complex) eigenvalues of $Df(x^*)$ are not on the unit sphere: $|\lambda| \neq 1$. For a hyperbolic equilibrium, the transformation to the linear part $y_{t+1} = Df(x^*) \cdot y_t$ with $y = x - x^*$ does not change the stability:

- $x^*$ is stable for $x_{t+1} = f(x_t)$ iff 0 is stable for $y_{t+1} = Df(x^*) \cdot y_t$.
- $x^*$ is asymptotically stable for $x_{t+1} = f(x_t)$ iff 0 is asymptotically stable for $y_{t+1} = Df(y_t) \cdot y_t$.

As for the continuous case, this allows the analysis of the stability of hyperbolic equilibria:

- $x^*$ is asymptotically stable iff $|\lambda| < 1$ for all eigenvalues $\lambda$ of $Df(x^*)$.
- $x^*$ is unstable iff $|\lambda| > 1$ for some eigenvalue $\lambda$ of $Df(x^*)$.

As an example we consider the discrete version of the logistic model: $N_{t+1} = N_t \exp(r(1 - \frac{N_t}{K}))$, where the system parameter $r > 0$ defines the birth rate and $K > 0$ is the capacity. We get the normalized model with $n_t = \frac{N_t}{K}$:

$$n_{t+1} = n_t \exp(r(1 - n_t))$$

We have two equilibria: $x^* = 0$ and $x^* = 1$. The derivation of the map $f(n) = n \exp(r(1-n))$ yields $f'(0) = \exp(r) > 1$ and $f'(1) = 1 - r$. Therefore $x^* = 0$ is an unstable equilibrium and $x^* = 1$ is an asymptotic stable equilibrium for $0 < r < 2$.

Every continuous dynamic system can be discretized: The time-discrete analogue of the continuous system $\dot{x} = X(x)$ is $x_{t+1} = x_t + X(x_t)$. For $f(x) := x + X(x)$ We have the following properties:

- $x^*$ is an equilibrium of the continuous system iff it is an equilibrium of the discrete system.
- The Jacobian matrix is $Df(x^*) = E + DX(x^*)$, where $E$ us the unit matrix.
- Each Eigenvalue of $Df(x^*)$ is of the form $1 + \lambda$ for an Eigenvalue $\lambda$ of $DX(x^*)$.
- If $|1 + \lambda| < 1$ then the real part of $\lambda$ is negative.
- The asymptotic stability in the discrete system implies the asymptotic stability in the continuous system.
- But an (asymptotic or neutral) stable equilibrium in the continuous system can become unstable in the discrete system.

### *C. Markov processes*

Another macro-level modeling method is based on Markov processes [2].

A Markov process is a stochastic process that fulfills the Markov property: The next state change (time of state transition, destination state) depends on the current state only, not on the state history of the stochastic process. The time can be chosen either discrete or continuous, and also the state space can be chosen either discrete or continuous. A Markov process with discrete state space is also called Markov chain. A Markov chain with discrete time $T = \mathbb{N}$ can be described by a discrete set $S$ of states and probabilities for the state transitions $p_{ij}$ for $i, j \in S$. The probabilities $p_{ij}$ might be independent of the time (homogeneous Markov chain) or dependent on the time (inhomogeneous Markov chain). Stable equilibria of discrete-time Markov chains are absorbing states, which can directly be identified by observing a diagonal element of value 1 within the transition probability matrix $\mathbf{P} = (p_{ij})_{i,j\in S}$. If a unique steady-state probability vector $\boldsymbol{\pi}$ exists, it can be determined by solving the system of equations $\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{P}\,, \sum_{i\in S} \pi_i = 1\,$. A continuous-time Markov chain can be described by a discrete set $S$ of states and rates for the state transitions $q_{ij}$ for $i, j \in S$. The rates $q_{ij}$ might be independent of the time (homogeneous Markov chain) or dependent on the time (inhomogeneous Markov chain). Stable equilibria can directly be identified by observing a row of zeros within the CTMC's transition rate matrix $\mathbf{Q}$. If a unique steady-state probability vector $\boldsymbol{\pi}$ exists, it can be obtained by solving $\mathbf{0} = \boldsymbol{\pi}\mathbf{Q}\,, \sum_{i\in S} \pi_i = 1\,$.

## III. Micro-level modeling

While macro-level modeling abstracts from the entities in the system and looks only at the variables of interest, micro-level modeling looks inside each entity: The behavior the the entities and their interactions have to be described. For the micro level modeling, we use the methods of [4].

### *A. Graph representation of system*

The topology of a system can be modeled by a directed graph $G$, where $V$ is the set of vertices and $K$ is the set of directed edges. Each node $v \in V$ represents one entity of the system and each edge represents a communication channel between two entities. The behavior of each entity $v \in V$ can be described by a stochastic automaton $a_v$:

- The automaton receives local input $x_w \in A$ from predecessor nodes $w \in pred(v)$.
- At each point of time the automaton has an internal state $s \in S_v$.
- The automaton decides nondeterministically about the change of state and about the local output to the successor nodes. Probability distributions can be used to describe the outputs and the state transition function of each stochastic automaton.

The influence of the environment on the system can be modeled by special vertices (external nodes) in the graph.

As an example consider some cars on a highway, which have an ambient intelligence (AmI) device. Information about the current position and the current speed is transmitted by the device to the devices of the other cars. Based on the information sent by the other cars, the device can give hints to the driver (e.g. "traffic jam ahead" or "slow down"). Each node of the graph represents one car. In this model, the edges of the graph change dynamically: There is an edge between two cars iff they are close to each other, such that communication is possible. The internal state of the automaton contains some information like speed and position. At each point in time, the edge between two nodes contains the information sent by the AmI device. While the behavior of the AmI devices is deterministic, the change of the state can be modeled stochastic, because the behavior of a human being is usually not predetermined.

When we consider the global view on the system at a point of time, then we see a current local state inside each automaton and a current value on each edge, which is transmitted from one node to another node. Such a global view is called *configuration*. It represents a snapshot of the system.

To analyze the behavior of a system, we initialize it at time $t_0 = 0$ by choosing a start configuration $c_0 \in \Gamma$, where $\Gamma$ is the set of all possible start configurations (with a given probability distribution). Then the automata produce a sequence $c_0 \to c_1 \to c_2 \to \dots$ of configurations during the run of the system. Since the automata and the initialization are not deterministic, the sequence $c_0 \to c_1 \to c_2 \to \dots$ is not uniquely determined by the system, but it depends on random events. So for each time $t \geq 0$, we have a random variable $\mathrm{Conf}_t$, which describes, with which probability $P(\mathrm{Conf}_t = c)$ the system is in a given configuration $c$ at time $t$.

Because of the exponential growth of the global state space with increasing number of entities, an analytical analysis of the system is usually impossible. Therefore, micro-level models are often analyzed with computer simulations: In each simulation run, pseudo random number generators are used to produce samples for the random variables in the model. This can be done for specifying the initialization (the pseudo random number generator produces an initial configuration) and for the state change in each step in the automata (the pseudo random number generator produces the next internal state and the local output to the successor nodes). Then the behavior of the system is deduced from the results of the simulation runs.

For the example of traffic with AmI devices, such simulation runs can be done for the analysis of different strategies (which advice should the AmI device give to the driver?) to achieve different goals (e.g. low size of traffic jam, high throughput, high safety, ...).

## B. Cellular automata

Cellular automata (CA) work on a set of cells. At each point in time, each cell has an internal state. The (deterministic or stochastic) change of the state of a cell depends on the current state and on the states of its neighbors. For an infinite cellular automaton of dimension $n$ the set of cells is $C = \mathbb{Z}^n$ and for a finite cellular automaton of dimension $n$ the set of cells is $C = \mathbb{Z}_L^n$ with periodic boundaries (e.g. cells on a torus). Let $Q$ be the set of all possible states of a cell.

Let us now consider a one-dimensional CA. For a cell $i \in C$ and a point in time $t \geq 0$ the current state in cell $i$ is denoted by $s(i,t)$. The state transition can be represented by a (deterministic or stochastic) function $s(i,t+1) = f(s(i-r_l,t), s(i-r_l+1,t), \dots, s(i+r_r,t))$, where $r_l$ and $r_r$ are left and right radius of the rules. This local evolution operator $f$ has $n = r_l + r_r + 1$ parameters. For a stochastic CA, the image of local evolution operator can be seen as a discrete random variable with values in $Q$. The local evolution operator induces a global evolution operator $F$ on the set $Q^C$ of all configurations of the cellular automaton: The next configuration $S_{t+1}$ is the result of the global evolution operator $F$ applied on the current configuration $S_t$:

$$S_{t+1} = F(S_t)$$

For 2-dimensional CA different lattices and different neighborhoods possible. For example the lattice can be a rectangle with a Moore neighborhood or von Neumann neighborhood (see Fig. 3).

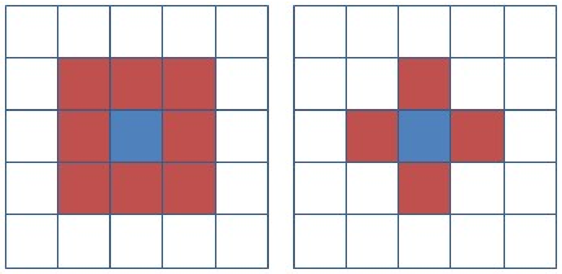

Figure 3. Moore neighborhood (left) and von Neumann neighborhood (right)

Elementary CA are the easiest form of CA. They have dimension $n = 1$ and two states $Q = \{0,1\}$. The state 1 means "active cell" and 0 means "empty cell". The radius is $r_l = r_r = 1 = r$. The neighborhood of a cell consists of three cells, so the local evolution operator has the form $f : Q^3 \to Q$. There are $2^8 = 256$ different elementary local rules. Such a rule can be represented by a table. One example is given in table I. This table defines for each combination of three old states the new state. The 8 binary values can be represented as a decimal value (code of the rule): $10111000_2 = 184$. Rule 184 can be used as a model for cars driving from left to right, where an active cell means, that a car is inside the cell. If the next cell is free, the car drives to the next cell, otherwise it stays at the same cell.

Starting at time $t = 0$ with a random initialization of the cells, the rule is applied in each step to compute

| 111 | 110 | 101 | 100 | 011 | 010 | 001 | 000 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |

Table I
RULE 184

the next valuation of the cells. For the analysis of the behavior of a cellular automaton we usually are interested in the properties of the global state for $t \to \infty$. Especially questions like "Is there a global structure emerging from a random initialization?" are of main interest. In the case of the rule 184 it can be shown that for a car density $\rho \leq \frac{1}{2}$ in the finite lattice $\mathbb{Z}_L$ all traffic jams will disappear: Every car is surrounded by two empty cells. For $\rho = \frac{1}{2}$ the resulting structure is an oscillation with period 2: Each cell is active in one step and empty in the next step. For $\rho \geq \frac{1}{2}$ all empty spaces are reduced, such that every empty cell is surrounded by two active cells.

For other rules, where the number of objects in the system is not constant, other questions might also be interesting. For example, does the density $\rho(t)$ of the objects converge for $t \to \infty$? Different analysis methods (e.g. mean field approximation or local structure theory) can be found in [1].

Note that a cellular automaton can be seen as a special case of the model described in Section III-A: The edges in the graph are the links in the neighborhood and the communication from one node to another node is just its internal state.

## IV. QUANTITATIVE MEASURES FOR ANALYSIS AND EVALUATION OF SELF-ORGANIZING SYSTEMS

In the last years, much research has been done in the topic of self-organizing systems. There is no generally accepted meaning of self-organization. Some typical features of self-organization are autonomy, emergence, self-maintenance, adaptivity, decentralization, and optimization.

The topic of self-organizing systems can be considered with the following goals:

1) *Analysis and evaluation of self-organizing properties in a given system.*
To achieve this goal, the system is analyzed with respect to the features mentioned above: Is the system fully autonomous? If not, how much external control is needed to fulfill the given task? Does a special structure emerge from the local interactions between the entities? These and other questions have to be answered to decide, which self-organizing features the system has.
2) *Optimization of a given system with respect to some specified self-organizing properties.*
There are many possibilities to make changes to a given system (e.g. change a system parameter or some local rules, introduce some new components with additional features, etc.) to achieve this goal. The impacts of such changes can be measured in the real system or analyzed in a model (either analytically or with simulations).
3) *Design and engineering of a new self-organizing system.*
In this task, design criteria have to be specified, such that the system fulfills the desired self-organizing features. Here we can use mathematical models to compare different design decisions with respect to some specified self-organizing features.

All three goals have the same requirement: We need quantitative measures that can be applied to a specified system (or to a model of a system) and which yield information about the contained self-organizing properties. Some quantitative measures have been developed in the recent years:

- *Autonomy*
How much control data from external entities are needed to keep the system running?
- *Emergence*
How many globally coherent patterns are induced by local interactions?
- *Target orientation*
Is the high level goal, that the system designer had in his mind, reached by the system?
- *Adaptivity*
Is the high level goal still reached after changes in the environment?
- *Resilience*
Is the high level goal still reached after unexpected impacts on the system (e.g. break down of some nodes, attacks by an intruder)?
- *Global state awareness*
How much information does a single node have about the global state (averaged over all nodes)?

Each of these measures yields a value in the interval $[0, 1]$, where $1$ means, that the self-organizing property is fully satisfied, while $0$ means, that it is not satisfied at all. Some of the quantitative measures are based on fitness functions, which have to be defined by the system designer (what is the goal of the system and what are the good/bad configurations?), other quantitative measures are based on the statistical entropy $H(X)$, which measures the amount of information of a given random variable $X$.

In [3] the level of emergence is defined by measuring dependencies in the communication. In some systems, it may happen, that some patterns or properties appear in the system as a whole, but do not appear in the single components. For a point in time $t \geq 0$ the *level of emergence at time $t$* is defined by

$$\varepsilon_t = 1 - \frac{H(\mathrm{Conf}_t\,|_K)}{\sum\limits_{k \in K} H(\mathrm{Conf}_t\,|_{\{k\}})}$$

In this definition, $H(\mathrm{Conf}_t\,|_K)$ measures the entropy of the values on the edges (communication between entities) in the configuration at time $t$ without taking account of the internal states of the automata. The information of all edges is compared to the information contained in each single edge. If at the current point in time $t \geq 0$ there are large dependencies between the values on the single edges (which can be seen as patterns), the level of emergence is high: $\varepsilon_t \approx 1$. If the values of nearly all edges are independent, there will be no pattern, so the level of emergence is low: $\varepsilon_t \approx 0$. Therefore the map $t \mapsto \varepsilon_t$ measures the dependencies occurring during the whole run of the system.

Note that in literature the definition of emergence is not unique. There also exist some work (see e.g. [6]), where emergence is defined as an unexpected decrease in relative algorithmic complexity.

## V. Application: Slot synchronizing in wireless networks

In this section we show, how the modeling and evaluation methods of the sections III and IV can be used for a slot synchronizing algorithm in wireless networks. Other applications of the presented micro-level modeling and macro-level modeling methods can be found in sections II and III-B

In the slot synchronization algorithm [7] we consider a wireless network, where time is divided into slots for communication. There is no central clock, which defines when a slot begins. The nodes try to synchronize the slots in a decentralized manner. At each point in time, each node is in one of four different states (see Fig. 4):

- In the transmission state, the node transmits a pulse to its neighbors to indicate the beginning of a slot.
- In the listening state, the node can receive and decode pulses from its neighbors and it adjusts its phase function $\phi$ according to these pulses. The listening state ends, when the threshold $\phi_{max} = 1$ is reached.
- In the waiting state and in the refractory state, the node does nothing.

The length of an uncoupled cycle is $2T$ with $T > 0$.

In the corresponding micro-level model, the graph describes the topology of the wireless network: $V$ is the set of wireless nodes and $K$ is the set of edges representing the communication channels between the nodes. The alphabet for communication is $A = \{0, 1\}$, where $1$ is a pulse and $0$ is no pulse. The current state $s \in S_v$ of a node $v \in V$ contains information about

- the current value of the phase function $\phi \in [0, 1]$,
- the current position in the cycle $\gamma \in [0, 2T]$,
- the decoding delays of received pulses.

The graphs of $\gamma$ and $\phi$ in dependency of the time are shown in Fig. 5 and Fig. 6.

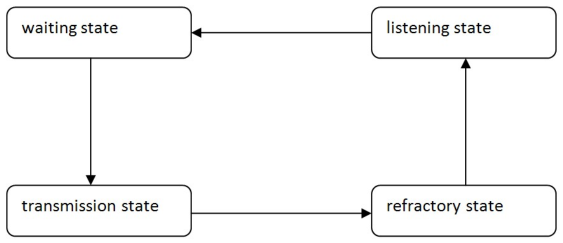

Figure 4. Four states in the slot synchronization algorithm

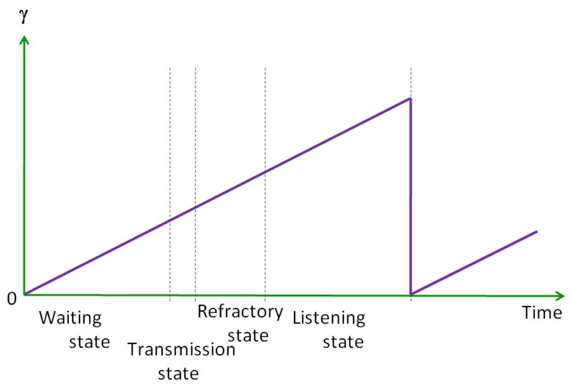

Figure 5. $\gamma(t)$ in the slot synchronization algorithm

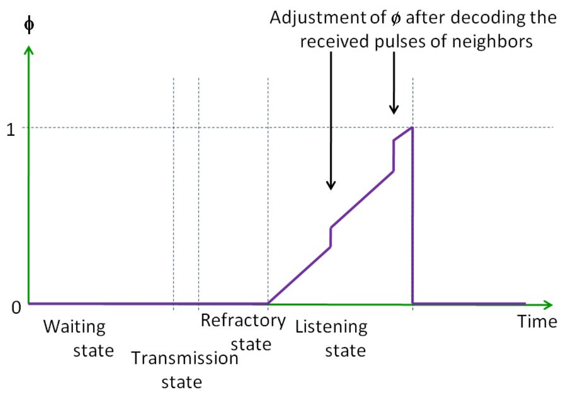

Figure 6. $\phi(t)$ in the slot synchronization algorithm

Each node $v$ stores some information $D_w$ about the pulse received from a neighbor $w \in V$ in its internal state: When a firing pulse is received at the object $v$ from another object $w$ in the listening state then the value $D_w$ is initialized with a constant $T_{dec}$. After the transmission of the pulse is finished, $D_w$ decreases during the time. When $D_w$ reaches the value $0$, then the decoding of the pulse is finished, and the phase function $\phi$ is adjusted by adding a value $\Delta\phi$. Negative values for $D_w$ are used to indicate that the value is irrelevant, since the phase function has already been adjusted according to the received pulse of w. Also in the other intervals (waiting state, transmission state, refractory state) a negative value is used. The state space of the automaton $a_v$ is $S_v = [0,1] \times [0,2T] \times (-1, T_{dec}]^V$. The output map of the automaton $a_v$ yields the value 1 during the transmission state and 0 otherwise. The system is autonomic, because no external control is needed, so the quantitative measure for autonomy yields $\alpha_t = 1$ for all points in time. The synchronization of the objects can be seen as an emergent pattern: it is a global structure in the system, which is induced by the local interactions. After the synchronization is finished, the level of emergence $\varepsilon_t$ is nearly one, where the exact value of $\varepsilon_t$ depends on the number of nodes in the system: The larger the system, the more dependencies exist in the communication. The level of emergence (after completing the synchronization) in dependency of the number of nodes is shown in Fig. 7.

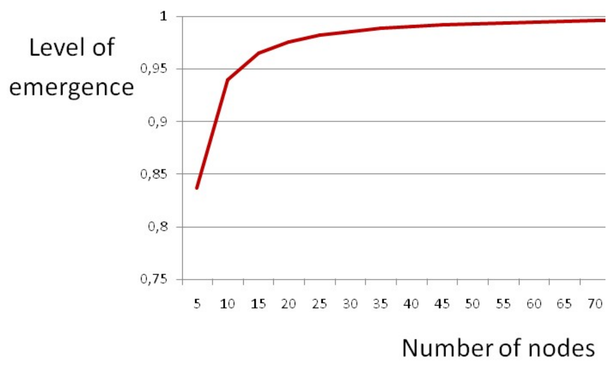

Figure 7. Level of emergence after synchronization in dependency of the number of nodes

## VI. Conclusion

Mathematical modeling can be used for a wide variety of systems: Technical systems, biological systems, physical systems, and many more. Quantitative measures can be used in models for the analysis, evaluation and optimization of existing systems and for the design and engineering of new self-organizing systems.

## Acknowledgment

This work 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).

## References

[1] N. Boccara, *Modelling Complex Systems*, ser. Graduate texts in contemporary physics. Springer, 2004.

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

[3] R. Holzer, H. De Meer, and C. Bettstetter, “On Autonomy and Emergence in Self-Organizing Systems,” in *IWSOS 2008*, ser. LNCS, K. A. Hummel and J. P. Sterbenz, Eds., vol. 5343. Springer, 2008.

[4] R. Holzer and H. De Meer, “Quantitative modeling of self-organizing properties,” in *Proc. of the 4th Int’l Workshop on Self-Organizing Systems (IWSOS 2009)*, ser. Lecture Notes in Computer Science (LNCS), T. Spyropoulos and K. Hummel, Eds., vol. 5918. Springer-Verlag, 2009, pp. 149–161, the original publication is available at www.springerlink.com.

[5] R. Holzer, P. Wuechner, and H. De Meer, “Modeling of self-organizing systems: An overview,” *Electronic Communications of the EASST*, vol. 27, pp. 1–12, 2010.

[6] J. L. Dessalles and D. Phan, “Emergence in multi-agent systems:cognitive hierarchy, detection, and complexity reduction,” Society for Computational Economics, Computing in Economics and Finance 2005 257, November 2005. [Online]. Available: http://ideas.repec.org/p/sce/scecf5/257.html

[7] A. Tyrrell, G. Auer, and C. Bettstetter, “Biologically inspired synchronization for wireless networks,” in *Advances in Biologically Inspired Information Systems: Models, Methods, and Tools*, ser. Studies in Computational Intelligence, F. Dressler and I. Carreras, Eds. Springer, 2007, vol. 69, pp. 47–62.