On Modeling of Self-organizing Systems*

Richard Holzer
University of Passau
Innstrasse 43
94032 Passau
Germany
holzer@fim.uni-passau.de

Hermann de Meer
University of Passau
Innstrasse 43
94032 Passau
Germany
demeer@fim.uni-passau.de

ABSTRACT

A goal of computing and networking systems is to limit administrative requirements for users and operators. A technical systems should be able to configure itself as much as possible to increase the usability. This leads to the design of self-organizing systems. Self-organizing systems emerge as an increasingly important area of research, not only for computer networks but also in many other fields. For analyzing properties of complex systems, a mathematical model for these systems may be useful. Whether a model with discrete time or with continuous time fits better, depends on the properties of the system and which analysis should be done in the model. In this paper we give a comparison between discrete and continuous models and we give a formal definition for modeling continuous complex systems. Then this theory is applied to model slot-synchronization in wireless networks.

Categories and Subject Descriptors

I.6.5 [**Model Development**]: Modeling methodologies

General Terms

Modeling Systems

Keywords

Self-Organization, Mathematical modeling, Systems

1. INTRODUCTION

In computing and networking systems, self-organization becomes more and more important. Self-organization cannot only reduce administrative requirements and configuration work, but a self-organizing system should also be able to detect and correct failures automatically if possible. Another trend in networked systems is to distribute the network control and data among the entities of a network such that the system becomes more independent of centralized servers and control instances.

A non-technical overview of self-organizing systems can be found in [6]. Self-organization can be seen as the increase of coherence or as the decrease of statistical entropy. The main properties of self-organization are:

- Autonomy: Nearly no external control is needed for the system.
- Emergence: Local interactions induce the creation of globally coherent patterns.
- Adaptivity: Changes in the environment have only a small influence on the behavior of the system.
- Decentralization: The control of the system is not done by a single entity or by just a small group of entities, but by all entities of the system.

Other definitions and properties of self-organizing systems can be found in thermodynamics [10], synergetics [5], information theory [11] and cybernetics [13], [1], [2], [7]. A good overview about modeling complex systems can be found in [3]. For modeling discrete self-organizing systems see [8].

In this paper section 2 gives a comparison between continuous modeling and discrete modeling with respect to self-organization. Section 3 proposes a formal method for the modeling of continuous systems. In section 4 we apply the definitions of section 3 to model the algorithm of [12] for slot-synchronization in wireless networks. Section 5 concludes this paper.

2. DISCRETE VERSUS CONTINUOUS

When we design a model for a complex system, then we first have to check, which parts of the system should be modeled discrete, and which parts should be modeled continuous. There are four major parts, where this decision has a large impact on the design and on the behavior of the model. They are: time, object set, states of objects, interaction.

In a model with discrete time, we only consider a finite or countable number of steps in time, so a model with discrete time usually is used, when behavior is event driven, i.e. for the time between two events there is no need to model anything of the system, so only the (finite or countable) events appear in the model. A model with continuous time usually

*The presented work has been supported by the EU projects Euro-NF (NoE, FP7, ICT-2007-1-216366) and AutoI (STREP, FP7, ICT-2007-1-216404).

AUTONOMICS 2008, September 23-25, Turin, Italy

DOI 10.4108/ICST.AUTONOMICS2008.4671

is used, when changes in the system do not occur in form of events, but every time.

In a model with a discrete object set, we only consider a finite or countable number of objects, which are interacting with each other. If the number of objects is uncountable, a continuous object set can be used in the model. Also if the number of objects is finite, but very large, such that an analysis of a system with a discrete object set would be very difficult (e.g. if the objects are the elementary magnets of a metal plate), then it might be more convenient to use a continuous object set.

In a model with discrete states of objects, at each point of time each object is in one of a finite or countable set of states. The future behavior of an object (change of state, interaction to other objects) depends on the state of the object. States can for example be used to store information about the input, which the objects got from other objects or from the environment in the past. Even if there are uncountable many states of the object in the real world, it might be better to use a discrete set of states if only finite or countable information is needed to describe the behavior of the object. Only if such a discrete set does not suffice to store the needed information, a continuous set of states might be the better choice for the model.

In a model with discrete interaction, each object can interact at each point of time with only finite or countable many other objects. If an interaction occurs not only between finite or countable many objects, but everywhere in the space (e.g. gravitation force of a planet), then the interaction is continuous. In a model with discrete interaction, the interaction can be seen as the usage of communication channels: Each communication channel connects two objects, and an interaction can be described as the transfer of data through this channel. A continuous interaction can also be seen as a force or an impression of one object to other objects.

While section 3 of this paper uses continuous methods for modeling systems, [8] describes how discrete methods can be used to model systems. In [8], graphs are used to describe which objects can communicate with which other objects. Each node in the graph represents one object and each edge represents one communication channel. The behaviors of the objects are described by stochastic automatons. Since the time is discrete in [8], each automaton has a clock, which defines, when the successor state will be computed. Modeling with discrete methods has the advantage that a concept of the information theory can be used to quantify the information of data: *Entropy*. For a discrete random variable X taking values from a set W the entropy [4] is $H(X) = -\sum_{w \in W} P(X = w) \log_2 P(X = w)$. The entropy is a measure for the average number of bits, which are needed for an optimal encoding of the information. The entropy can also be seen as a measure of uncertainty: high entropy for the system means that nearly no information is known in advance and a low entropy means that nearly all information is known in advance. With this concept, many things of the systems can be measured:

- How much information is contained in the whole system at time t?
- How much control data is needed from the environment to compute the next configuration during the run of the system?
- How much information is processed by one object in a single step?
- Which dependencies appear in the communication between the objects?

In discrete systems, the entropy can be used to answer these questions. In [8], formal definitions of the level of autonomy and level of emergence are given. Since the measurement of information with the entropy can only be applied to discrete systems, these definitions cannot be translated into the theory of continuous modeling. On the other way, continuous models are usually much smoother than the corresponding discrete models, because in a continuous model the whole behavior is described by differential equations, while in a discrete model each single step of the automatons must be described. It would also be very difficult to compute the entropy of a large system.

Concerning the example in section 4, the definitions of [8] also show that the system is autonomous and it has a high level of emergence. Adaptivity and decentralization have not been formally defined in [8].

3. MODELING CONTINUOUS SYSTEMS

In this section we give a mathematical definition for modeling systems. The time and the state set are modeled continuously. For the object set and the interactions, it depends on the system to be modeled, whether they should be modeled discrete or not. The definitions in this section can be applied for both cases. For modeling systems with discrete time and discrete state sets see [8].

A system consists of a set of objects V, which can interact with each other, i.e. at each point of time $t \in \mathbb{R}_0^+$ each object $v \in V$ can send information to other objects. These objects can be of technical nature (e.g. computers connected by a LAN), of biological nature (e.g. a colony of ants) or of other kind. At each point of time $t \in \mathbb{R}_0^+$ each object $v \in V$ of the system has a current state, which can change over the time. A state transition within an object depends on the current object state and interactions with other objects.

To describe also nondeterministic behavior of objects, we assume that we have a probability space $\Sigma = (\Omega, \mathcal{A}, P)$, where Ω is the set of all random events, $\mathcal{A}$ is a σ-algebra and $P : \mathcal{A} \to [0, 1]$ is a probability measure on $\mathcal{A}$. This probability space will be used to describe all random events of the objects occurring during the time. If each object $v \in V$ has its own probability space Σ_v, which is independent of the probability spaces of the other objects, then we can use the product space $\Sigma = \prod_{v \in V} \Sigma_v$. If there are independent probability spaces $\Sigma_{v,t}$ for each point of time $t \in \mathbb{R}_0^+$, then we can use the product space $\Sigma = \prod_{v \in V, t \in \mathbb{R}_0^+} \Sigma_{v,t}$. A *random map* between two sets A, B is a family $g = (g^\omega)_{\omega \in \Omega}$ of maps $g^\omega : A \to B$. We use also the notation $g : A \to B$ for random maps, i.e. g is seen as a map, which depends on the random event. Analogously, a random set $C = (C_\omega)_{\omega \in \Omega}$ is seen as a set, which depends on the random event. Now we give a formal definition of the mathematical model of systems.

DEFINITION 1. *Let $\Sigma = (\Omega, \mathcal{A}, P)$ be a probability space. A* continuous system *$\mathcal{S} = (V, S, A, \lambda, f, h)$ consists of*

- *a set V, where the elements of V are called* objects of the system*;*

- *a family $S = (S_v)_{v \in V}$ of sets, where each set S_v is a subset of a normed vector space; the elements of S_v are called* states of the object v*;*
- *a set A, which is called* alphabet*;*
- *a family $\lambda = (\lambda_v)_{v \in V}$ of random maps $\lambda_v : S_v \times V \to A$, where λ_v is called* output map of the object v*.*
- *a family $f = (f_v)_{v \in V}$ of random maps $f_v : A^V \times S_v \to S_v$, where f_v is called* change map of the object v*.*
- *a family $h = (h_v)_{v \in V}$ of random maps $h_v : A^V \times S_v \to S_v$, where h_v is called* hop map of the object v*.*

An initialization *of the system $\mathcal{S}$ is a random variable $I = (I^\omega)_{\omega \in \Omega}$ with $I^\omega = (I_v^\omega)_{v \in V} \in \prod_{v \in V} S_v$ for all $\omega \in \Omega$.*[1]

A family $(s_v)_{v \in V}$ of random maps $s_v : \mathbb{R}_0^+ \to S_v$ is called behavior of $\mathcal{S}$ *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) $\{t \in \mathbb{R}_0^+ \mid s_v$ *is not differentiable in* $t\}$ *is a discrete random set,*

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

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

The behavior of the system describes for each object $v \in V$ and each point of time $t \in \mathbb{R}_0^+$ the current state $s_v(t)$ of the object v. The initialization defines the states at time $t = 0$. The output map λ_v defines the value $\lambda_v(s, w)$ that the object v sends to the object w, when v is in the state s. The change map f_v describes how the state $s \in S_v$ continuously changes during the time. f_v can be seen as the derivation $\dot{s}$ of the state with respect to the time in dependency of the local inputs from other objects: For each object $w \in V$, the object w can send some data $x_w \in A$ to the object v, so the object v receives at the current point of time $t \in \mathbb{R}_0^+$ a family $x = (x_w)_{w \in V} \in A^V$ of values. If the current state of the object v is s, then $f(x, s)$ describes the direction, in which the state is currently moving: $f(x, s) = \dot{s}$. The hop map h_v describes the changes of the state, whenever the state is not continuous, i.e. $h_v(x, s)$ can be seen as the value that has to be added to s to get the new state after receiving the local input x, so we have the new state $s_{\text{new}} = s + h_v(x, s)$. Since the time is not discrete, we have no "successor state" like in the discrete case (see [8]), so s_{new} is only the limit from the right: $s_{\text{new}} = \lim_{p \searrow t} s_v(p)$, where t is the current point of time.

Note that the output map λ_v cannot depend on the current input of the other objects, because if we would define λ_v as a random map from $A^V \times S_v \times V$ to A, then this would mean that the value that is sent from an object $v \in V$ to another object $w \in V$ at time $t \in \mathbb{R}_0^+$ depends on the value that is sent from the object w to v at the same time t and vice versa. Obviously this would lead to problems, so λ_v can only depend on the current state but not on the current input values from other objects. But it is still possible to model the situation that v sends a new value $b \in A$ to the object w after it received a certain value $a \in A$ from w by changing the state with the hop map h_v. Then the output map λ_v can be used to send the new value to w, since λ_v depends on the state.

It could also be possible to restrict the interactions, e.g. if some objects in the real world are too far away, such that a direct communication is not possible. This could be modeled by a graph, where the edges describe the possible communication channels. In [8] this has been done for discrete modeling. But since many systems in the real world (e.g. ant colonies) are not static, the graph could change during the time. This is the reason why we do not use a graph for the modeling. The problem of missing communication channels can be solved by using a special symbol $null \in A$ for "no communication".

From Definition 1, it can easily be seen from the hop map h_v, in which situations the behavior s_v of the object v is not continuous:

LEMMA 2. *Let $\mathcal{S}$ be a system, s be a behavior and $t \in \mathbb{R}_0^+$. The following conditions are equivalent:*

- *s_v is continuous at time t.*
- *$h_v((\lambda_w(s_w(t), v))_{w \in V}, s_v(t)) = 0$*

4. MODELING SLOT-SYNCHRONIZATION

In this section we apply the definitions of the previous section to model an algorithm for self-organized slot-synchronization in wireless networks [12]. In such a network, we have some objects which can communicate with the other objects. For this communication, the time is divided into time slots. Since there is no central clock, which defines when a slot begins, the objects need to apply a slot synchronization. An algorithm for this slot synchronization is proposed in [12]. It is based on the model of pulse-coupled oscillators by Mirollo and Strogatz [9]. The clock of each object is described by a phase function ϕ which starts at 0 and increases over time until it reaches the threshold value $\phi_{th} = 1$. The object then sends a "firing pulse" to its neighbors for synchronization. After receiving the firing pulse, the other objects adjust their own phase functions by adding $\Delta\phi := (\alpha - 1)\phi + \beta$ to ϕ, where $\alpha > 1$ and $\beta > 0$ are constants.

In [12] delays (e.g., transmission delay, decoding delay) are introduced into this algorithm. Let $T > 0$ be a constant. The duration of an uncoupled period (i.e. if no pulses are received from other objects) is $2T$. Now this period is divided into four states (see Figure 1). Let $\gamma \in [0, 2T]$ be a time instant. Then the object is in a

- waiting state, if $\gamma \in [0, T_{wait}) =: I_{wait}$
- transmission state, if $\gamma \in [T_{wait}, T_{wait} + T_{Tx}) =: I_{Tx}$
- refractory state, if $\gamma \in [T_{wait} + T_{Tx}, T_{wait} + T_{Tx} + T_{refr}) =: I_{refr}$
- listening state, if $\gamma \in [T_{wait} + T_{Tx} + T_{refr}, 2T) =: I_{Rx}$

where the constants are defined as follows:

- T_{Tx}: Delay for the transmission of a value.

[1] We also use the notation $I \in \prod_{v \in V} S_v$.

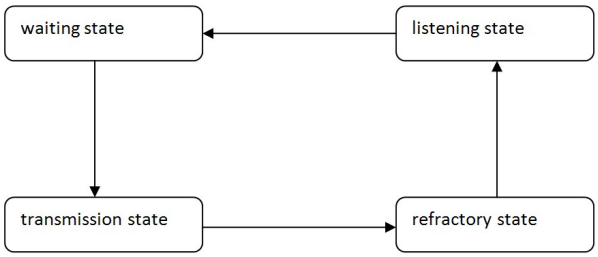

Figure 1: State diagram

- T_{wait}: Waiting delay after the phase function reached the threshold. The transmission of the firing pulse begins after this delay. The waiting delay is defined by $T_{wait} = T - (T_{Tx} + T_{dec})$, where T_{dec} is the decoding delay, i.e. the time that an object needs to decode a received value.

- T_{refr}: Refractory delay after the transmission of the firing pulse to avoid an unstable behavior.

Define $T_{Rx} = 2T - T_{wait} - T_{Tx} - T_{refr}$. Then T_{Rx} is the duration of an uncoupled listening state. This is the only state, in which firing pulses from the neighbors can be received and decoded, and the phase function is changed only during the listening state. We assume that each of these durations $T_{wait}, T_{Tx}, T_{refr}, T_{Rx}$ is less than T.

The continuous system $\mathcal{S} = (V, S, A, \lambda, f, h)$ contains a (finite) set V for the objects of the system. For the alphabet we use $A = \{0, 1\}$ to describe the firing pulses between the objects (0 means not firing and 1 means firing).

For an object $v \in V$, we store the following data in the current state $s \in S_v$:

- the current value of the phase function $\phi \in [0, 1]$,

- the position in the cycle $\gamma \in [0, 2T]$,

- some information $D = (D_w)_{w \in V}$ about the decoding delays for the received pulses. When a firing pulse is received at the object v from another object w during the interval I_{Rx}, then the value D_w is initialized with 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 ϕ is adjusted by adding $\Delta\phi = (\alpha - 1)\phi + \beta$. 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 $I_{wait}, I_{Tx}, I_{refr}$ we use a negative value for D_w to indicate the irrelevance.

Therefore, a state $s \in S_v$ is a triple $s = (\phi, \gamma, D)$.

Now let us consider the output map λ_v. Only the interval I_{Tx} is used for the transmission, so during this interval, the output value 1 is sent to all other objects $w \in V$. During the intervals $I_{wait}, I_{Rx}, I_{refr}$ there is no pulse, so the output value 0 is sent to all other objects.

The change map f_v describes the derivation $\dot{s}$ of the state s. Let us first consider the intervals $I_{wait}, I_{Tx}, I_{refr}$: During these intervals, the phase function stays constant 0, so $\dot{\phi} = 0$. Also the value D_w for $w \in V$ can stay constant at a negative value, so $\dot{D}_w = 0$. Only the variable γ is changed. Since γ is the time elapsed since the beginning of the cycle, we have $\dot{\gamma} = 1$. During the interval I_{Rx} we have a different change of the state: In this interval, the phase function ϕ increases uniform (if no pulses arrive from other objects) until the threshold is reached. For an uncoupled system, the threshold ϕ_{th} should be reached at the end of the listening period, so during the interval of length T_{Rx} the phase function ϕ grows linearly from 0 to 1, which implies $\dot{\phi} = \frac{1}{T_{Rx}}$. The value γ still grows with gradient $\dot{\gamma} = 1$ like above. During a pulse $x_w = 1$ of another object $w \in V$, the value D_w stays constant at T_{dec}, so we have $\dot{D}_w = 0$. After the end of this pulse, D_w decreases with the gradient $\dot{D}_w = -1$.

The hop map h_v describes the changes of the state, whenever the state is not continuous. In $I_{wait}, I_{Tx}, I_{refr}$, the values ϕ and D_w for $w \in V$ are constant and γ is continuous, so in this case the hop map is 0. Now consider the interval I_{Rx}. The value γ is still continuous. For a pulse from another object w, the value D_w must be set to T_{dec}. Since the hop map describes only the value that has to be added to the old state, this can be done by adding $-D_w + T_{dec}$ to the old value D_w to get the new value T_{dec}. After the decoding delay also an adjustment of ϕ has to be done. If the threshold is not reached by this adjustment, this is done by adding $\Delta\phi$ to the current value of ϕ for each pulse that has been decoded, i.e. for $D_w = 0$. For $D_w > 0$, the pulse has not yet been decoded, so the adjustment of ϕ need not be done yet. $D_w < 0$ indicates irrelevance (either the adjustment has already been done or no pulse has been received from w), so also in this case no adjustment to ϕ has to be done. After reaching the threshold $\phi_{th} = 1$, the phase function must be set to 0, and also γ must be initialized to 0 to begin the new cycle. In this case we use the hop map to get into the state $(0, 0, (-1)_{w \in V})$.

Now we give a formal definition of these maps λ_v, f_v, h_v. Let $S_v = [0, 1] \times [0, 2T] \times (-\infty, T_{dec}]^V$ and $(\phi, \gamma, D) \in S_v$. Definition of the output map λ_v:

- For $\gamma \in I_{wait} \cup I_{refr} \cup I_{Rx}$ define $\lambda_v(\phi, \gamma, D, w) = 0$.

- For $\gamma \in I_{Tx}$ define $\lambda_v(\phi, \gamma, D, w) = 1$.

Definition of the change map f_v:

- For $\gamma \in I_{wait} \cup I_{Tx} \cup I_{refr}$ define $f_v(x, \phi, \gamma, D) = (0, 1, (0)_{w \in V})$.

- For $\gamma \in I_{Rx}$ define $f_v(x, \phi, \gamma, D) = (\frac{1}{T_{Rx}}, 1, (x_w - 1)_{w \in V})$.

Definition of the hop map h_v:

- For $\gamma \in I_{wait} \cup I_{Tx} \cup I_{refr}$ define $h_v(x, \phi, \gamma, D) = (0, 0, 0)$.

- For $\gamma \in I_{Rx}$ let $\phi' = \Delta\phi \cdot |\{w \in V \setminus \{v\} \mid D_w = 0\}|$ with $\Delta\phi := (\alpha - 1)\phi + \beta$. For $w \in V$ let $D'_w = -D_w + T_{dec}$ for $x_w = 1$ and $D'_w = 0$ for $x_w = 0$. Let $h_v(x, \phi, \gamma, D) = (\phi', 0, D')$ for $\phi + \phi' < 1$ and $h_v(x, \phi, \gamma, D) = (-\phi, -\gamma, (-D_w - 1)_{w \in V})$ for $\phi + \phi' \geq 1$.

The following Theorem shows that we have successfully modeled the system of [12].

THEOREM 3. *For each initialization I, the system described above has exactly one behavior $s = (s_v)_{v \in V}$. The value γ of*

the current state $s_v(t)$ for an object v runs cyclic through the intervals $I_{wait}, I_{Tx}, I_{refr}, I_{Rx}$. After the end of the first cycle, the states $(\phi, \gamma, D) = s_v(t)$ of the behavior have the following properties:[2]

1. *In each cycle, γ starts at 0 and grows linearly with $\dot{\gamma} = 1$ until the end of the cycle and then restarts at 0.*
2. *During the intervals $I_{wait}, I_{Rx}, I_{refr}$, the object v does not send a pulse.*
3. *During the interval I_{Tx}, the object v sends a pulse.*
4. *During the intervals $I_{wait}, I_{Tx}, I_{refr}$, the phase function ϕ is constant 0.*
5. *During the intervals $I_{wait}, I_{Tx}, I_{refr}$, the value D_w is constant -1.*
6. *If the interval I_{Rx} starts at time t (i.e. $\gamma = T_{wait} + T_{Tx} + T_{refr}$) then ϕ is continuous in t.*
7. *During the interval I_{Rx}, if $\phi = 1$, then the current cycle ends and the next cycle starts.*
8. *During the interval I_{Rx} with $0 < \phi < 1$, the phase function ϕ is not differentiable at time t iff $s_v(t - T_{dec})$ was a listening state and a pulse from another object w ended at time $t - T_{dec}$.*
9. *During the interval I_{Rx}, if the phase function ϕ is differentiable, then $\dot{\phi} = \frac{1}{T_{Rx}}$.*
10. *If a pulse from another object $w \in V$ ends during the interval I_{Rx} of the object v and $s_v(t+T_{dec})$ is still in the listening state, then the phase function ϕ is adjusted by adding $\Delta\phi$ to ϕ (if the new value is smaller than 1) at time $t + T_{dec}$. This is done for each object $w \neq v$, for which the pulse ends at time t. If the new value is greater or equal to 1, then the current cycle ends and the next cycle starts.*
11. *During the interval I_{Rx} the value D_w decreases linearly with gradient -1 during the time, where no pulse arrives from w.*
12. *During the interval I_{Rx} the value D_w is constant T_{dec} during the time, where a pulse arrives from w.*

Proof. A formal proof of this theorem would be out of scope of this paper, but it is straight forward to check the following conditions:

- Every behavior of $\mathcal{S}$ satisfies the properties 1-12, because of the definitions of λ_v, f_v and h_v of $\mathcal{S}$.
- The random maps s_v for $v \in V$ are uniquely determined by the initialization and the properties 1-12, i.e. for a given initialization, there is only one family of random maps $s = (s_v)_{v \in V}$ satisfying 1-12, because these properties describe the whole course of the random maps.
- For each random map s_v that satisfies 1-12, the properties (B2)-(B5) of Definition 1 are also satisfied.

After verifying these conditions, the assertion of this theorem can be easily deduced. □

[2]If we assume a meaningful initialization, these properties are also satisfied during the first cycle.

The simulation results in [12] show that during the run of the system, groups of synchronizations are built, i.e. inside each group we have good synchronization (each object of the group fires at nearly the same time like the other objects of the group), and if we wait long enough, then there are only two groups left firing T time units apart from each other.

This system satisfies the four main properties of self-organization:

- The system is autonomic, because no external control is needed.
- The synchronization of the objects is emergence, since this is a global property of the system, which is induced by the local interactions.
- The system is adaptive to small changes in the environment (adding more objects, removing some existing objects, etc.).
- The control of the system is completely decentralized.

5. CONCLUSION AND FUTURE WORK

For analyzing properties of complex systems, the real world system can be transformed into a mathematical model. In this paper we compared continuous modeling with discrete modeling and we gave a formal definition for modeling continuous complex systems. Then this theory has been applied to model the slot-synchronization algorithm of [12].

While a level of autonomy and the level of emergence can be defined formally in discrete systems (see [8]) with the concept of entropy, it is still an open problem, how they can be defined formally for continuous systems. Also a formal definition of the level of adaptivity and the level of decentralization is left for future work.

6. REFERENCES

[1] W. R. Ashby. *Principles of Self-organization*, chapter Principles of the Self-organizing System, pages 255–278. Pergamon, 1962.

[2] S. Beer. *Decision and Control: The Meaning of Operational Research and Management Cybernetics*. John Wiley & Sons, Inc., 1966.

[3] N. Boccara. *Modeling Complex Systems*. Springer-Verlag, 2004.

[4] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, 2nd edition, 2006.

[5] H. Haken. *Self-organizing Systems: An Interdisciplinary Approach*, chapter Synergetics and the Problem of Selforganization, pages 9–13. Campus Verlag, 1981.

[6] F. Heylighen. The science of self-organization and adaptivity. In *in: Knowledge Management, Organizational Intelligence and Learning, and Complexity, in: The Encyclopedia of Life Support Systems, EOLSS*, pages 253–280. Publishers Co. Ltd, 2003.

[7] F. Heylighen and C. Joslyn. Cybernetics and second order cybernetics. *Encyclopaedia of Physical Science & Technology*, 4:155–170, 2001.

[8] R. Holzer, H. de Meer, and C. Bettstetter. On autonomy and emergence in self-organizing systems. In *IWSOS 2008*. Springer, 2008. Submitted.

[9] R. Mirollo and S. Strogatz. Synchronization of pulse-coupled biological oscillators. *SIAM Journal of Applied Mathematics*, 50:1645–1662, 1990.

[10] G. Nicolis and I. Prigogine. *Self-Organization in Non-Equilibrium Systems: From Dissipative Structures to Order Through Fluctuations*. Wiley, 1977.

[11] C. R. Shalizi. *Causal Architecture, Complexity and Self-Organization in Time Series and Cellular Automata*. PhD thesis, University of Wisconsin-Madison, 2001.

[12] A. Tyrrell, G. Auer, and C. Bettstetter. Biologically inspired synchronization for wireless networks. In F. Dressler and I. Carreras, editors, *Advances in Biologically Inspired Information Systems: Models, Methods, and Tools*, volume 69 of *Studies in Computational Intelligence*, pages 47–62. Springer, 2007.

[13] H. von Foerster. *Self-Organizing Systems*, chapter On Self-Organizing Systems and their Environments, pages 31–50. Pergamon, 1960.