Towards Self-Coordinating Ubiquitous Computing Environments

Franz J. Rammig
Heinz Nixdorf Institute
University of Paderborn
Paderborn, Germany
{franz@upb.de}

Abstract We are surrounded by an enormous amount of microprocessors. Their quantity outnumbers the human population by a factor of more than three. These microprocessors enable most technological artifacts to become intelligent *"things that think"* and a majority of these intelligent objects will be linked together to an *"Internet of things"*. This omnipresent virtual *"organism*" will provide ubiquitous computing to an amount which goes far beyond all presently existing systems. To master this emerging virtual organism, completely new paradigms of operation have to evolve. In this paper we present our vision of establishing self-coordination as the dominant paradigm of operation of future ubiquitous computing environments. This vision is looked at from four different points of view. First of all techniques to model self-coordinating distributed systems in an adequate manner is discussed. Then the principle of self-coordination is applied to individual intelligent objects. In a next step such objects have to be arranged in a networked manner. Again the potential of self-coordination, now applied to communication infrastructures is studied. Finally self-coordination is applied to next generation interfaces between human beings an artificial ones. In this paper we do not attempt to provide a complete discourse of the area. Instead of this we try to illustrate the four aspects mentioned above by proper examples.

Keywords: Self-coordination, Organic Computing, ant-colony algorithms

1 Introduction

In the world of information technology it is no longer the computer in the classical sense where the majority of IT applications are executed; computing is everywhere. More than 20 billion processors have already been fabricated and the majority of them can be assumed to still be operational. These microprocessors enable most technological artifacts to become intelligent "*things that think"*. At the same time virtually every PC worldwide is connected via the Internet. This combination of traditional and embedded computing creates an artifact of a complexity, heterogeneity which is rarely manageable by classical means. Metaphorically speaking, the emerging ubiquitous computing environment may be treated as an organism made up of computers, networks, system software, applications, and, most importantly, human beings.

This virtual organism as it exists today is still in a state of adolescence. Each of our technical artifacts with a built-in processor can be seen as a "*Thing that Thinks*", a term introduced by MIT's Thinglab. It can be expected that in the near future these billions of *Things that Think* will become an "*Internet of Things*", a term originating from ETH Zurich. This means that (using the above metaphor) we will be constantly surrounded by a virtual *"organism"* of *Things that Think*.

In order to deal with such kinds of ubiquitous computing environments, novel principles, methods, and tools to design such a virtual organism and its constituents are needed. A very promising solution may be seen in handing over a large amount of design activities to the system itself. This means that a large portion of decisions which traditionally take place in the design phase, now are handed over to the operational phase. Even more important, these decisions no are made by the system itself, based on information about its environment and its own nature. To a certain degree, the objects and the resulting virtual organism develop themselves. The bare size of the virtual organism to be handled makes any attempt of a centralized solution senseless. Instead of this, self-coordinating principles have to be established on the majority of the individual objects constituting the entire system, and even more important on the interconnection structure. In the sequel various aspects of introducing self-coordination as the basic paradigm of future ubiquitous computing environments are discussed.

In any scientific discipline it is hardly possible to deal with objects which can not be modeled. Therefore a well adapted modeling paradigm for self-coordinating, highly distributed systems is a major step towards establishing the paradigm of self-coordination. This aspect will be discussed in section "Modeling". Some main principles are explained before the basic ideas are illustrated by an example.

If the concept of self-coordination is to be applied to the entire field of ubiquitous computing environments, both the individual constituents and the network made out of theses objects should follow this paradigm. In section "Self-Coordinating Objects" basic principles are discussed, how future intelligent object can be empowered to act in a self-coordinating manner. Again an example of some already completed work is used to illustrate this principle.

An even higher potential of self-coordination arises if the global interaction of such objects is envisioned. By analyzing solutions found in biological systems, especially in ant colonies, approaches can be looked for, how such communication structures and problem solving strategies can be handled by means of self-coordination. This topic is covered by section "Self-Coordinating Networked Objects". Here again, results from previous work is used as example to illustrate the basic approach.

Finally, these intelligent objects that form a virtual organism are, in and of themselves, of no value. They need to be able to serve human needs. In the end, artificial assistants offering intelligent services have to come into existence. To achieve this level of service provisioning, a discipline that may be characterized as *Anthropomatics* (a term originating from the Univ. of Karlsruhe and used by the German Organic Computing initiative) has to evolve. Communication of human beings is not restricted to a bare intellectual level. In addition, emotions play an important role. If intelligent artifacts shall be accepted by humans as adequate communication partners and friendly servants, this aspect of emotions has to be considered as well. I.e. an intelligent artifact or some virtual "organism" must be able to express emotions (without having ones, of course) and to recognize emotions of human communication partners. In section "Self-Coordinated Anthropomatics" this aspect is exemplified using some previous work.

2 State of the Art

Modeling of self-coordinating distributed systems

Some theoretical work has been published characterizing the emergence of global behavior patterns based on local rules. A prominent, currently extensively studied area is given by game-theoretical approaches initiated by Koutsoupias and Papadimitriou [12]. However, it is still unclear how local rules can be designed for a given target behavior, what their possibilities/restrictions in comparison to global rules are, and how to deal with real-time limitations.

Concerning description techniques, methods, and tools needed for the practical construction of such systems, software engineering has developed model-based development and formal analysis techniques for analysis and transformation of requirements and system design down to executable code. However, they are more or less only applicable before the system is deployed – in no way do they address the inherent dynamics in problem-solving and reconfiguration of structures and algorithms or scalability properties needed here. The necessity to model systems where the constituents act strictly individually and only based on local information (i.e. in absence of any kind of global control), is reflected by the basic principle of Petri nets and derivatives.

The initiatives Autonomic Computing [11], Organic Computing [20], and parts of the European Complex Systems Initiative [5, 7] are interesting attempts to attack the mentioned challenges. These initiatives use inspirations from biological systems, transferring such principles into the engineering domain. However, both initiatives rarely deal with the coordination paradigm and systems of highly dynamic structure are investigated only marginally.

Self-coordinating objects

Self-coordinating objects can be found in user-oriented IT-systems, IT-based games [KRP04], logistics, robotics, just to mention some examples. In the IT-area first systems following the self-x paradigms of Autonomic Computing have been reported in the area of servers and information systems.

The various *Robocup* contests may serve as interesting experimental environments where self-coordinating "intelligent" agents have to be developed [17]. These agents have to act autonomously and in a team-oriented manner together with their team mates. Methods from various research traditions like artificial intelligence, cognition, smart sensors, mechatronics, ad-hoc networks, etc. have to be combined to obtain good results.

Various results concerning evolutionary virtual creatures have been published, e.g. [19, 13, 14]. Systems are reported that allow the creation and the evolution of virtual creatures that adapt to varying environments or that compete for rare resources. Even physical the realization of such creatures has been reported [14].

Self-coordinating Networked Objects

Networking became an important area of systems-oriented research (e.g., self-organization as typically found in MANETs [15]). New systems concepts such as cooperative networking or in-network processing have appeared, mostly in sensor networks (e.g., distributed source/network coding). Although algorithm theory has resulted in excellent solutions in the area of routing (e.g. [16]), self-coordinating techniques are applied as well. Based on Marco Dorigo's pioneering work [6] various attempts have been made. In [9], e.g., the authors directly apply this technique to a reactive, hop-by-hop routing protocol. Approaches to build overlay networks with the aid of ant colony algorithms have been reported as well, e.g. [8]. Following this paradigm, highly adaptive and emergent network construction and optimization techniques can be established.

Self-coordinating Anthropomatics

In an aging population the development of interactive artificial assistants is of great relevance. In this field, one can observe the convergence of different research topics (e.g. machine learning, statistics, reasoning, neuroscience, computer science, and engineering) into a new field called *Anthropomatics*. Artificial assistants interact constantly with their human users; therefore they will be cognitive systems that can reason and learn from their experience. As they are aware of their own capabilities and reflect on their own behavior they can be seen as instances of Organic Computing. To make the interaction with humans more natural, these artificial assistants should be able to express emotions and to recognize emotions of their communication partners. This aspect has been identified long ago [2]. More recent publications address this aspect from various points of view [1, 3].

3 Aspects of Self-Coordinating Ubiquitous Computing

3.1 Modeling of self-coordinating distributed systems

The volatility of the upcoming new pervasive and ubiquitous systems, the lack of global control, their emergent behavior, and their stringent requirements on dependability and security require a kind of new *"System Theory for Self-coordinating Systems"* and, in particular, an improved understanding of coordination paradigms.

Most of today's methods are limited to essentially static cases where the behavior of the composed system can be predicted beforehand. In the future, however, we can instead endow generic components with more or less strict rules on how to behave, how to interact with other components, and even how to change these rules if necessary. These components need to act in a self-coordinated way to reach application goals in a volatile environment.

Numerous completely new research questions arise to address these issues. As the entire research, basic and applied one, depends on adequate modeling techniques, in this paper only the modeling aspect is addressed.

Models and Modeling Techniques

Model-based design and development is widely accepted as the most promising way to master complexity in today's system design. Certainly it will remain indispensable for the new types of systems. Unlike for traditional static systems, modeling realistic behavior of these systems is challenging. Most of the established techniques relay on the existence of global states. In highly distributed systems and especially in such ones that are based on local decisions without global control, global states do not exist at all or at least of no importance at all. Therefore modeling techniques are needed that are communication centric, that are not dependent on global states and that are based on local transitions. At the same time a profound mathematical foundation is required.

Modeling techniques are necessary for expressing models and their dynamics at different abstraction levels based on sound semantic foundation and mathematical elegance. This technique needs a "calculus" to enable model manipulation, e.g., refinement between abstraction levels or verification. Multi-dimensional modeling in an integrated manner, including non-functional and mobility aspects is an important part of such techniques. This requires languages operating on meta-level, integrating models written in different languages. A new dimension in the definition of language semantics is a consequence of this approach, i.e. the semantics of the meta-model depend on the semantics of the underlying models. Using such modeling techniques, realistic model**s** have to be constructed for the dynamic volatile behavior of specific classes of systems. They must support application-oriented system as well as design and analysis of algorithms. They should provide new ways to understand and abstract from properties of such systems, such as profiling methods that become the basis for generating formal models suitable for solid experimental and theoretical analysis, e.g., models for user movement that are not only amenable to simulation but also to mathematical analysis. The most demanding property, however, of such a modeling technique is the ability to describe self-modification in an elegant and easy to understand way.

Example: Self Modifying Extended Predicate/Transition Nets

The main properties requested for modeling self-coordinating ubiquitous computing environments can be summarized as:

- Communication centric,
- Local control,
- Absence of global state,
- Asynchronous,
- Modeling support for complex systems,
- Heterogeneity,
- Support of self-modification.

It can be observed that the first 4 properties are already present in case of Petri nets. Higher order Petri nets like Predicate/Transition nets even support the description of complex and heterogeneous systems, at least when hierarchy is added, as done by work at our group. Therefore it seems to be a good advice to follow the path of higher order Petri nets as the basis for the needed modeling technique. The main open gap is self-modification. As it will be shown, even this gap can be closed in an elegant manner.

Petri nets are bipartite directed graphs $PN = (P, T, f, g)$ augmented by a marking M and firing rules. The Petri net graph consists of a finite set of places P, a finite set of transitions T, directed edges $f \subseteq P x T$ from places to transitions and $g \subseteq T x P$ from transitions to places. Places model conditions. For this purpose they may be marked by tokens. Driven by specific firing rules a transition can fire based on the local marking of those places it is directly connected with. By firing, the marking of these places is modified.

In the case of High-Level nets the tokens are typed individuals. The other net components are annotated accordingly: places with data types, edges with variable expressions and transitions with a guard and a set of variable assignments. Now a transition can fire only if the formal edge expressions can be unified with actually available tokens and this unification passes the guard expression of the transition. By firing, the input tokens are consumed and calculations associated with the transition are executed. By this new tokens are produced that are routed to output places according to variable expressions annotating the output edges of the transition.

Pr/T–Nets are very powerful, offering the modeling power of Turing machines. To support easy modeling of highly complex systems in our research group we additionally added a hierarchy concept that combines hierarchy over places (following Harel's hierarchy concepts for *StateCharts* [10]) and over transitions (in accordance with the concepts of *Structured Nets* by Cherkasova/Kotov [4]).

Finally we added description means for timing to Pr/T–Nets. By this we obtained an elegant, adequate and powerful modeling means that serves as foundation of our entire work on design systems for distributed embedded systems. However up to now this modeling technique is adequate only for structurally static systems. Structurally dynamic systems demand for some further extensions that are to be described in the sequel.

In the case of static systems the resulting system can be modeled in advance. In the case of a dynamically reconfigurable one, only the generating system of a set of potentially resulting systems can be provided. Obviously this is an even more demanding task. Resulting systems are modeled by extended Pr/T–Nets in our case. By reasons of intended elegance and in order to achieve a self-contained concept we decided to use an extension of these modeling means to describe the generation process as well [18].

For realizing dynamic reconfiguration, we propose to apply the mechanism of net refinement – usually only used during the specification of a system – at run-time. Technically, changes to the refinement of components are associated to the firing of certain transitions. For this purpose, we allow to annotate transitions with a rule for the refinement of other nodes in the surrounding Petri net specification.

Usually it is not reasonable to specify rules applicable to arbitrary transitions. Therefore, restrictions on the application of a rule can be specified. On the one hand, a scope may be specified, that is a subnet in which the transformation may take place. On the other hand, attributes of the component, whose refinement is changed, can be specified in the guard of a transition. That way it is possible for instance, to specify precisely to which transition the transformation should be applied by specifying its fully qualified path name.

Using refinements for net modifications combines two advantages. On the one hand, Petri net refinements are a thoroughly elaborated concept for developing a specification in a well-structured way. They allow for defining transformations that preserve certain relevant properties of the specification. On the other hand they are powerful enough for describing a variety of dynamic modifications. However, in some cases it is necessary to extend the limited set of refinement operations. We therefore offer a generic approach in addition to the one described above. We allow annotating transitions with rules as they are used in High-Level replacement systems. Annotating an arbitrary transition *TMod* of a Petri net *N* with this rule specifies that, after firing of *TMod*, a subnet *Left* of *N* is replaced by the net *Right*. As for the transitions annotated with refinement rules, also for transitions with replacement rules the firing rule is extended. In addition to the standard checking for fireability of transitions it is checked whether the left hand side of the rule is part of the current Petri net. During transition firing, the rule is executed in addition to the standard firing actions.

3.2 Self-Coordinating Objects

Numerous IT-based objects are enhanced by built-in intelligence. This means that a sense-decide-act loop is included in such systems. By exploring the environment using smart sensor functionalities decisions can be made how to behave in the current situation. These decisions depend not only on actual environmental information but also on a certain history of both, input from the environment and internal state. The latter means that the system has to have certain reflective capabilities. Based on this reflective knowledge, information from the environment including previous environmental reaction on activities of the system, and some global knowledge the system is able to learn and by this self-optimize its behavior. Systems of this kind are emerging in the areas of smart user interfaces, in gaming, and in the field of mechatronics.

Possible applications are robotics, driver assistant systems, advanced vehicle dynamics control, traffic systems with coordinated vehicle control, energy management, and route planning. Cognitive actuators with integrated smart sensors and inherent intelligence will take over major parts of the sense-decide-act loop. An intelligent actuator will be able to supervise itself and to take care of its health. All these constituents of embedded software have to be seen as a whole. Today's reconfigurable hardware and the upcoming developments in this field allow making late decisions whether control algorithms have to be implemented in HW or SW. This distribution even may change at runtime, dependent on the actual value of various resources.

Example: Self-Emerging Virtual Creatures

Imagine a virtual creature that can adapt itself to become able to match a certain objective. There is some research into this direction, e.g. as reported in [3, 13, 14]. In the latter case (the Golem project) even physical realizations of such virtual creatures are looked at. In a thesis at our lab, Tim Schmidt concentrated on an automated design procedure for such creatures. The basic principle can be characterized by an evolutionary algorithm where all constituents of such a virtual creature (morphology, dynamics, sensing, and control) are object to this evolutionary process. Even more than other work, Tim Schmidt provides a powerful simulation environment that allows observing and evaluating the evolutionary process easily during the design process.

The virtual creatures considered are formed by a tree structure of cubic solids connected by joints. Number, shape, and dimensioning of the solids are object to evolution. The joints may allow any degree of movement in all three axles. Joints may be located at any location on the surface of the solids. The only actuators are motors inside the joints. They may provide movements at any speed into any direction within an assignable limit. Sensors, too, may be located at any location on the surface of solids. They just sense contacts between solids and other solids or between solids and the environment.

The control system of these creatures is a distributed one. It consists of a set of Limited Recurrent Neural Networks. There is a local controller per joint. It controls the rotation of the two connected solids relatively to each other. The controllers observe the values gathered by the relevant sensors. In addition the various controllers communicate in order to provide their control services. So a completely self-coordinating system is obtained.

Construction of the virtual creatures takes place using a genetic algorithm. The morphology of creatures is coded using *Binary String Covering*. The leaf nodes of a creature (all creatures are tree-structured) are coded by constants while the inner node coding is calculated by application of appropriate functions. Based on such a gene coding, the usual genetic operations like mutation and crossover are applied. The resulting creatures are evaluated concerning the respective objective. Promising creatures form the next generation of the population.

3.3 Self-Coordinating Networked Objects

The most dramatic change in systems' nature comes up if services are provided by **networks** in the most general sense. In history any kind of traffic (i.e. networking) was one of the most important driving factors in enhancing culture and technology. Therefore it can be expected that omnipresent and seamless networking of technical artifacts will have a comparable impact on modern technology. The service-providing entities are usually considered as autonomous and cooperating towards a common goal, but competition and selfishness of various kinds is also present. There is a wide range of "helper" disciplines engaged to improve such services; for example, optimization theory has found its application in virtually every kind of network. In most cases, a central resource planning is usually not feasible, nor is a static planning – perturbations in network structure and requests have to be accounted for and corrected.

Services and networks need to be composed. Networks usually do not exist in isolation; rather, they come in contact with each other. A team of soccer playing robots [17] may serve as an example. The natural computing architecture of future ubiquitous computing systems is that of a distributed, concurrent, and communicating set of processing nodes (sensors, actuators, controller units). To meet the demands of high reliability and hard real-time processing of such complex heterogeneous networks, integrated software/hardware architectures have to be explored together with optimally matching operating system services. Self-coordination is mandatory for resource-efficient design and management of such distributed controller architectures. Operating systems themselves will become network-based. I.e. the whole set of offered services will no longer be provided by each single instance of the OS by a cluster of instances as a whole. The same techniques to migrate application software from one computing node to another can also be applied to the OS. This turns such a network-based OS into a volatile, self coordinating artifact.

Example: Self-Coordinating Operating Systems / Communication Structures

Ordinary operating systems (OS) are forced to provide all services that might be asked for by any application as the set of possible application over the OS's lifetime is not predictable. In embedded systems, usually exactly these services are provided that are really needed by the set of (a priori known) applications. This saves a lot of resources, especially concerning memory. In the case of self-coordinating systems, however, the set of possible applications is no longer fixed and cannot be predicted off-line. In order to avoid the overhead of general purpose operating systems, techniques of self-adaptation, now applied to the OS are used. An extreme approach into this direction is Paderborn University's NanoOS. This OS is especially tailored towards swarms of tiny nodes, e.g. sensor networks. The basic idea of NanoOS is to use an entire swarm of instances running on top of a swarm of nodes as the service provider. This means that a broad band of services is available, however not necessarily entirely on a specific node. Rarely requested services may be distributed sparsely over the network. When such a service is requested and it is not available at the location of request it may be activated remotely on another node of the respective cluster. A cluster in this context is defined as the set of OS instances that provides the entire set of services.

Of course the distribution of services over the network is crucial. In the case of NanoOS ant colony algorithms are applied to approach an optimal distribution of services. Whenever a service is requested remotely all nodes on the path from the requesting node to the providing one is marked by *"pheromone"*. Services now have a tendency to move into the direction of the highest "pheromone" concentration.

Similar techniques are used to provide a self-coordinating communication structure in between the nodes of such a swarm. Ant colony algorithms are used to construct a less dense (and therefore less power consuming) backbone net. Again special ant colony algorithms are used to adapt the network dynamically to any kind of distortion. Shortest paths can be found using Dorigo's "classical" algorithm simulating the nest – food source behavior of ants.

3.4 Self-coordinating Anthropomatics

The term *anthropomatics* describes a scientific field that uses methods of computer, electrical, and mechanical engineering to develop models of interaction of humans with artificial agents, artifacts, servants, or assistants. Future technical systems will learn about the user and the environment without the user's explicit input to the machine. Recognition of the intention of the user by analyzing prior actions and his/her emotions together with the generation of intuitive feedback will constitute more enjoyable man-machine interaction. Methods for natural communication, emphasizing higher-level cognitive functions have to be applied. The coordinated combination of these techniques will allow humans to communicate in a situation-specific and context-sensitive manner with artificial assistants that are endued with cognitive capabilities.

The recognition of speech can be enhanced by the extensive use of vision to analyze gestures and postures of the user. To enhance haptic communication with cognitive systems, advanced sensors will have to be designed. A flexible electronic skin, e.g., could give robots an almost-human sense of touch. The processing of the input of olfactory sensors will allow a further step into total immersion technologies. The resulting vicarious interactive participation in activities and the carrying out of physical work will bring benefits to a wide range of users. Examples include emergency and security services, entertainment and education industries, and those of restricted mobility such as the disabled or elderly.

Example: Expression and Recognition of Emotions

Human beings communicate not only based on pure facts but also express and recognize emotions. This additional communication “channel” seems to be extremely efficient. Therefore any communication which does not include this aspect is experienced as un-natural by humans. A human-centric man – machine communication therefore should include the aspect of emotions. The problem, however is, that obviously a machine has no emotions at all. But this does not mean that a machine cannot express emotions. What is needed is an internal model of emotions and how they can be stimulated.

C-LAB, the cooperation between Paderborn University and SIEMENS, has developed MEXI. This is an artificial head that is able to express emotions like joy, demand for contacting humans, desire to play, anger. Of course MEXI does not have such emotions. But is carries an internal model of them and is able to express the respective modeled emotional state by properly moving mouth, eyes, ears, or the entire head. The internal model includes rules, how basic emotions are triggered by specific stimuli from the outside and on the other hand how they fade away by a decay process. Elementary emotions then are overlaid to result in complex ones.

Human emotions are not only triggered by external stimuli but also by cyclic moods or drives. In MEXI, this cyclic behavior is modeled as well. The resulting values provide an additional overlay level. By this an even more realistic expression of emotions is obtained.

If we have a model of emotions and if we know how they are expressed by facial patterns, this can also be used to recognize emotions of humans. From psychology the basics are known. Relatively few patterns are sufficient to conclude rather precisely which emotional state a person is in. In the case of C-LAB’s research, a combination of facial patterns and speech is used for analysis.

4 Summary

Self-coordination will play a dominant role in future IT systems which will consist of billions of interconnected things that think. Completely new approaches are needed to design and operate such intelligent IT environments. Substantial theoretical research is necessary to develop a “System Theory of Self-coordinating Systems”. On the other hand real systems of this kind have to be developed, built, and investigated. Both, self-coordinating objects and networks made out of them have to be studied. Finally the human being has to remain the central point of consideration. Preliminary results are already available. Some examples of such already existing approaches have been shown in the paper.

Acknowledgments This paper summarizes from the authors’ view point the results of intensive discussions with his co-workers and students. Especially the examples used originate from joint work with these persons. The author would like to thank especially Carsten Rust for his work on self-modification of enhanced Predicate/Transition nets, Christian Geiger and Tim Schmidt for their work on evolutionary virtual creatures, Tales Heimfarth and Peter Janacik for their results on self-coordinating operating systems and communication structures. Finally we would like to thank Bernd Kleinjohann and Natascha Esau for their pioneering work on emotion modeling and emotion recognition.

We also would like to thank our colleagues Wilhelm Dangelmaier, Friedhelm Meyer auf der Heide, Odej Kao, Holger Karl, Bärbel Mertsching, and Ansgar Trächtler for fruitful discussions about the presented research area.

References

1. R.C. Arkin, M. Fujita, T. Takagi, and R. Hasegawa: An ethological and emotional basis for human-robot interaction. In: Robotics and Autonomous Systems, vol. 42, no. 3, pp. 191-201, Elevier, 2003
2. J. Bates: The role of emotion in believable agents. CACM, 37(7), pp 122-125, 1992
3. C. Breazeal: Affective interaction between humans and robots. In: Proc of ECAL01, pp. 582-591, Prague, 2001
4. L. A. Cherkasova and V. E. Kotov: Structured Nets. In J. Gruska and M. Chytil, editors, LNCS 118, Springer Verlag, 1981.
5. The European Complex Systems Initiative, http://www.cordis.lu/ist/fet/co.htm , 2003
6. M. Dorigo and G. DiCaro: The Ant Colony Optimization Meta-Heuristic. New Ideas in Optimization, McGraw Hill, pp.11-32, 1999
7. The DELIS project, http://delis.upb.de/ , 2004
8. S. Goss, S. Aron, J.-L. DeNeubourg, and J. M. Pasteels: Self-organized Short-cuts in the Argentine Ant. Naturwissenschaften, 76, pp. 579-581, 1989
9. M. Günes, U. Sporges, and I. Buazizi: ARA – The Ant-Colony Based Routing Algorithm for MANETs. In: Proc. IWAHN’02, pp. 79-85, 2002
10. D. Harel. Statecharts: A visual formalism for complex systems. In: Science of Computer Programming, 8(3), pp. 231–274,June 1978.
11. The Vision of Autonomic Computing, http://www.research.ibm.com/autonomic/manifesto/ , 2003
12. E. Koutsoupias and C. H. Papadimitriou: Worst-case Equilibria. Symposium on Theoretical Aspects of Computer Science STACS 1999, Springer, 1999
13. M. Komusinski, Sz. Ulatowski: Framesticks: towards a simulation of a nuture-like world, creatures and evolution. In: Proc. Of the 5th European Conference on Artificial Life (ECAL99), Springer Verlag, pp. 261-265, 1999
14. Hod Lipson, Jordan B. Pollack: Automatic Design and Manifacture of Robotic Lifeforms, Nature 406, pp. 974-978, 2000
15. M. Mauve, J. Widmer, and H. Hartenstein: A Survey on Position-based Routing for Mobile Wireless Ad-Hoc Networks. IEEE Network 15(6), 2001
16. P. Papadimitratos, Z.J. Haas, and E.G. Sirer: Path Set Selection in Mobile Ad Hoc Networks. Proc. Of ACM Mobihoc’02, 2002
17. RoboCup Official Site. http://www.robocup.org/
18. C. Rust and F. Rammig: A Petri Net Based Approach for the Design of Dynamically Modifiable Embedded Systems. In Proc. IFIP DIPES2004, Kluwer, 2004
19. Karl Sims: Evolving 3D Morphology and Behaviour by Competition. Artificial Life, pp. 353-372, MIT Press, 1994
20. DFG SPP 1183 Organic Computing: http://www.organic-computing.de/spp , 2004