André Miede, Nicolas Repp, Julian Eckert, Ralf Steinmetz:
Self-Organization Mechanisms for Information Systems – A Survey. In: Association for Information Systems (AIS): Proceedings of the Fourteenth Americas Conference on Information Systems, Toronto, ON, Canada August 14th-17th 2008, Association for Information Systems (AIS), August 2008. ISBN 978-0-615-23693-3.

Self-Organization Mechanisms for Information Systems – A Survey

André Miede, Nicolas Repp, Julian Eckert, Ralf Steinmetz
Department of Computer Science
Technische Universität Darmstadt, Germany
{miede, repp, jeckert, steinmetz}@kom.tu-darmstadt.de

ABSTRACT

The increasing complexity of managing information systems encourages researchers, vendors, and enterprises to develop new approaches in order to cope with this challenge. This paper focuses on a survey of self-organization mechanisms and discusses the relevance for the improvement of information systems, i.e. by reducing administration labor. The survey includes the concepts of self-organization and emergence, consisting of successful natural mechanisms discovered in biology and other natural sciences. The purpose of the survey is to serve as a source of inspiration for adaptations and implementations of such mechanisms, eventually leading to better, self-organizing information systems.

Keywords

Self-Organization, Emergence, Enterprise Application Architecture, Service-Oriented Architectures

INTRODUCTION

The evolution of information systems (IS) and their technologies is closely related to the businesses they support. Modern, highly competitive economies require enterprises to adapt both quickly and continuously to changing circumstances and demands. Therefore, IS face the following requirements (Krafzig, Banke, and Slama 2005):

- *Simplicity*: the IS has to be understood and managed by the people working with it.
- *Flexibility*: changes to local details must not affect the overall system, i.e. must not break it.
- *Reusability*: despite high initial costs, the IS should be built from a repository of reusable software components.
- *Decoupling*: functional and technical logic should be strictly separated.

While these requirements might be met in initial releases of IS, subsequent changes seriously decrease the system's adaptability, as shown in Figure 1.

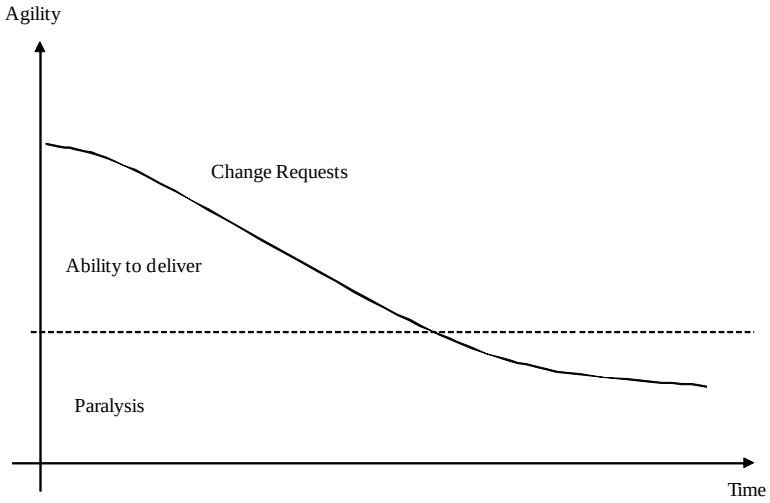

Figure 1: Ability-of-Adaptation-Curve (Krafzig et al. 2005)

Maintaining these requirements, e.g. by techniques such as loose coupling which can be achieved by Service-Oriented Architectures (SOA) (Papazoglou 2003), comes at the price of increased complexity of the IS. Therefore, many decisions that could previously be anticipated at the system's design time have to be deferred to the system's runtime (Kephart and Chess 2003). This makes complexity both a fundamental part and a major problem of IS which cannot be resolved totally but has to be reduced through the active management of the IS (Newcomer and Lomow 2004, Brooks 1995).

A possible solution to this problem is the concept of self-organization (SO), which spans over a wide variety of research fields. While SO is an active research topic for decades and even applications of the concept already found their way into IS research and related fields, this paper aims at giving an overview of SO mechanisms that are relevant for improving IS.

The goal of this paper is to make IS researchers aware of the possibilities SO offers and to provide ideas and insights from other academic disciplines such as biology that can be useful for IS.

The rest of this paper is structured as follows. In the next section we give an overview of related work that has dealt with SO and similar concepts before. The successive section defines both SO and emergence for the use in the context of IS. Thereafter, a variety of mechanisms for SO is presented and alternatives to SO are discussed. The paper closes with conclusions and an outlook on future research.

RELATED WORK

In this section we give an overview of both current research results from the area of SO as well as of major commercial initiatives related to SO.

Kephart and Chess (2003) introduced the concept of "Autonomic Computing" (AC). Its goal is self-managing computing systems with improved operation and maintenance capabilities. Their focus is highly technological and they discuss architectural building blocks and engineering problems of creating SO systems. (Due to Kephart's and Chess' affiliation to IBM, the work is tightly coupled to IBM's Autonomic Computing initiative, which is described in the next section.)

Di Marzo. Foukia, Hassas, Karageorgos, Mostéfaoui, Rana, Ulieru, Valckenaers, and Van Aart (2004) of the "Engineering Self-Organising Applications Working Group" compiled a variety of mechanisms that are used in nature to achieve SO behavior. Furthermore, they discussed a wide range of application examples for SO in the field of information technology (IT), e.g., computational Grids or business process infrastructures.

Nagpal (2003) discussed many mechanisms and primitives for SO and focused mainly on how to achieve robust behavior on a global level through the interactions of unreliable agents. The discussed mechanisms were discovered by exhaustive research in biology.

More recently, Liu and Tsui (2006) as well as Hinchey and Sterrit (2006, 2007; Sterrit and Hinchey 2005, 2006) revived the general discussion of SO concepts for information systems. They contributed both collections of SO mechanisms and potential application scenarios for them.

Very exhaustive and thorough discussions of SO and related concepts can be found in the work of the main researchers in this field, e.g. Kauffman (1996), Camazine, Deneubourg, Franks, Sneyd, Theraulaz, and Bonabeau (2003), Holland (1996), and Bonabeau, Dorigo, and Theraulaz (1999).

Commercial Initiatives

Apart from the academic interest in SO, its importance and potential for addressing complexity issues of IS is reflected by the involvement of a number of major IT companies. In the remaining part of this section, commercial SO activities spanning different layers of abstraction are outlined briefly.

As mentioned in the previous section, IBM's "Autonomic Computing" (AC) initiative (IBM 2008) includes groundbreaking work in this field of study and forms a basis for both further research and commercial applications (IBM 2006; Jacob, Lanyon-Hogg, Nadgir, and Yassin 2004; Ganek and Corbi 2003). AC, which is inspired by the human autonomous nervous system, can be seen as a technological form of SO, as technology is used to manage technology. It is mainly based on adaptable policies and control loops for both information gathering and reactions. A subset of the proposed capabilities was demonstrated by Jann, Browning, and Burugula (2003) at the infrastructure level, namely IBM's pSeries servers.

Furthermore, Intel Corp. has been working on introducing self-configuring capabilities into network services (Melcher and Mitchell 2004). Hewlett-Packard offers an "Adaptive Infrastructure" in its portfolio (Hewlett-Packard 2008). Since 2003, Microsoft Corp. is extending its "Dynamic Systems Initiative" (Microsoft 2008), enhancing its software such as Windows Vista and Windows Server with capabilities that support an adaptive IT infrastructure.

To conclude the related work overview, it is important to point out the many different fields of study SO shares it roots, techniques, and applications with. Examples include, but are not limited to the following: self-managing software, nature-inspired computing (NIC), autonomy-oriented computing, organic computing, adaptive computing, and swarm intelligence. (Liu and Tsui 2006; Hinchey and Sterrit 2006, 2007; Sterrit and Hinchey 2005, 2006; Hinchey, Sterrit, and Rouff 2007; Bonabeau et al. 1999)

In the next sections, we strive to both gather, update, and discuss the work of the above authors in more detail.

SELF-ORGANIZATION AND INFORMATION SYSTEMS

Much of what has been discovered about SO comes from the study of systems, i.e. physical, living, and social systems (Di Marzo et al. 2004). In this section, we collect and discuss important key elements of SO and their relevance for the use in IS.

Overview

To address the challenge of complexity in IS, the following three concepts are introduced and combined:

- *Self-Organization* is the capability of a system to make adaptations to itself without external control.
- *Emergence* is the result of an SO system using its mechanisms.
- *Mechanisms* are structures and means of an SO system to make a transition from one state into another, more ordered one.

These are no formal definitions but descriptions to make the relationships of the concepts to each other clearer (cf. Figure 2). Formal definitions and details on how these concepts work together follow in the next sections.

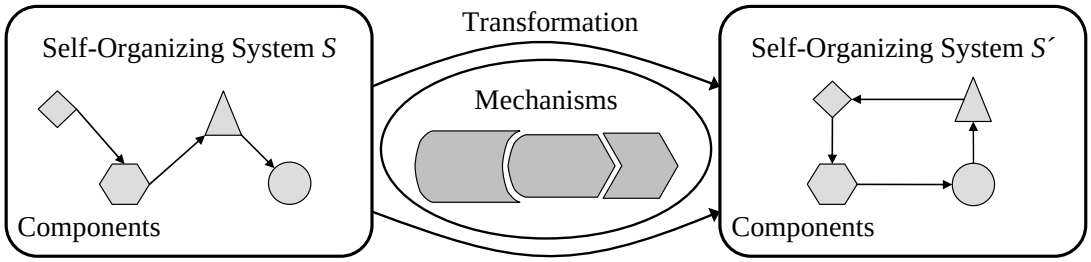

Figure 2: Relationship of Self-Organization Concepts

A simple example of the application of SO to IS is the usage of externally provided market data such as stock prices or currency rates in an internal information system. Using the mechanism "negative feedback", a concept which is discussed later on, an IS is able to detect that the flow of data is becoming irregular and therefore unreliable. The IS takes countermeasures by switching to another provider for market data who does not experience technical difficulties. Thus, the IS transforms itself without external intervention (for example from an administrator) into a new configuration, one that is again fully operational. To detail the example a bit more, we consider the above-mentioned IS to be designed according to the SOA paradigm, i.e. using services to model business processes. A business process for foreign money transfers would then – among others – be composed of a service to book an amount of money to an account. To calculate the correct amount, this service makes use of a service to provide currency exchange rates, e.g. available from external provider *A*. SO capabilities within the booking service could then detect the violation of Service-Level Agreements (SLA) and in response connect itself then to an alternative currency exchange rate service from provider *B*.

In the next sections we have a closer look at each of the introduced concepts separately.

Self-Organization

Self-organization (SO) is a concept that fascinates researchers in a wide range of fields for over 50 years. Goldstein (1999) gives an overview how SO and emergence relate to many other, seeming unrelated topics, for example Cybernetics, Fractal Geometry, or Artificial Intelligence. Furthermore, the interested reader can find a thorough discussion of the origins of SO in Shalizi's PhD thesis (2001), as we do not go into details about that here.

A wealth of SO concepts relevant for IS can be found in biological research. Whereas chemical and physical systems rely only on physical laws, biological and social systems offer the additional ingredient of genetically programmed behavior (Camazine et al. 2003). For the use in IS and a technological context in general, we introduce the following definition of SO, which follows definitions by Capra (1997), Goldstein (1999), Camazine et al.(2003), and Bonabeau et al. (1999):

Self-organization is the capability of a system to spontaneously create new structures and forms of behavior on the global level through internal interactions of its lower-level components and without external intervention.

SO contains subclasses of specialized capabilities which are generally referred to as "self-X" or "self-*" (Hinchey and Sterrit 2006). While there are many possibilities for "X" and "*", the four introduced by IBM (2006) and known as "CHOP" are the most famous ones:

- self-*c*onfiguration (adapt to changes in the system)
- self-*h*ealing (recover from detected errors)
- self-*o*ptimizing (improve the use of resources)
- self-*p*rotecting (anticipate and cure intrusions)

To get a better understanding how SO can be achieved and what its impacts are, we discuss the key characteristics and properties of SO systems in the following:

- *Dynamic*: The lower-level components interact continually to create and maintain spatio-temporal structures. (Camazine et al. 2003, Bonabeau et al. 1999, Capra 1997)
- *Nonlinearity and Emergent Properties*: In contrast to linear systems, the value of the whole cannot be calculated by the values of the parts' values alone. That is why the systemic, emergent properties cannot be understood by analysis – which is to decompose a system into its parts (reductionist approach) – but why the context of the global whole is necessary to understand the properties of the system's components. Thus, components are subject to two kinds of aggregation in SO systems:
    - *Mechanistic aggregation*, which focuses on how to model a complex system by reducing it to smaller and simpler parts.
    - *Holistic aggregation*, which is concerned with complex global behavior created from less complex components.

  (Holland 1996, Goldstein 1999, Camazine et al. 2003, Capra 1997)
- *Multistability*: Several stable states can coexist and the system has more attractors than just the state of equilibrium. (Goldstein 1999, Camazine et al. 2003, Bonabeau et al. 1999)
- *Bifurcation*: Small quantitative changes of the system's parameters amount to significant qualitative changes, i.e. sudden state transitions as amplifications of a random event. (Capra 1997, Goldstein 1999, Bonabeau et al. 1999)
- *Diversity*: The system's components are specialized and each fills a "niche" based on its interaction and processing capabilities. Ideally, removed components are replaced by other components taking over their interactions and work. (Holland 1996)

From this, we can see that SO systems feature characteristics that are not natively supported by many IS and which are therefore challenging to integrate. This opens the door to further research which is likely to improve IS and their use in modern enterprises.

Emergence

The concept of emergence and emergent behavior is tightly coupled to SO systems. For a brief history and discussion of its origins and development Goldstein's article (1999) in the "Emergence" journal's first issue is highly recommended.

In accordance with Goldstein's definition and our above understanding of SO, *we see emergence as the creation of new structures and new forms of behavior on the global level of a self-organizing system. These results arise through the lower-level, non-linear interactions between the system's components and are facilitated by self-organization mechanisms.*

Emergent phenomena exhibit certain common properties (Goldstein 1999):

1. *Dynamical*, emergence is not pre-defined but results in a SO system over time.
2. *Coherence and Correlation*, which unites multiple lower-level components into a global-level whole.
3. *Global Level*, due to the previous property, this is the level where emergence is observable.
4. *Radical Novelty*, emergence results in features that were not visible before in the system.

5. *Ostensive*, emergence is recognized by showing itself through its significant, radical new structures (cf. the previous property).

Concerning the use of emergent behavior in IS, especially properties 4 and 5 have to be considered carefully and have to be weakened, because they could lead to totally unpredictable and therefore uncontrollable phenomena – a situation which works in natural systems where other powers such as evolution are also at work, but which is highly undesirable for IS.

MECHANISMS TO ACHIEVE SELF-ORGANIZATION

In order for a SO system to transform itself into a new state and to show emergent behavior, the system and its components have to utilize certain mechanisms.

This section provides IS researchers with an overview of what nature is using successfully and to inspire adapted mechanisms in technological contexts. The following mechanisms are already abstracted from their original biological uses and use IS-specific nomenclature such as "component". For more information or biological and other examples please refer to the sources provided.

- *Building Blocks*: This is a mechanism usually already present in IS due to modern component-oriented software development approaches (Sommerville 2006) or the SOA paradigm. The purpose of building blocks is to introduce regularity, modularity, and therefore reusability into a complex system. This is valid at different levels of abstraction in the system, as the building blocks at one level should be reduced to the combinations and interactions of building blocks at the next lower level. (Holland 1996)
- *Decentralized Control*: Decision-making is transferred from a global, central instance to the components of the system themselves. It is therefore indirect and based primarily on local instead of global information. (Camazine et al. 2003)
- *Dense Heterarchies*: Unlike hierarchies where interaction is between the neighboring layers, heterarchies allow for the interaction between all components. There is no separate "upper management", the entirety of the components together constitutes the highest level of power in the system. This is closely related to the above-mentioned concept of decentralized control. (Camazine et al. 2003)
- *Positive Feedback (Amplification)*: This mechanism promotes changes in a system and is best described by the metaphor "snowball effect" (or more formally as "autocatalysis"). However, there is the danger of uncontrollable growth and eventual destruction inherent to this amplification. The next mechanism can be used in combination to control positive feedback. (Camazine et al. 2003, Bonabeau et al. 1999)
- *Negative Feedback*: To stabilize a system or to trigger corrective counteractions that maintain the status quo, negative feedback is used. It is often found in combination with positive feedback to limit and guide amplifications. (Camazine et al. 2003, Bonabeau et al. 1999)
- *Tagging*: To facilitate aggregation or the formation of boundaries in a system, tags can be used. Components or their interfaces are tagged with special information to achieve selective interaction, namely with other components that are looking for specific tags. Interaction is then only performed if the provided tag matches the required condition. (Holland 1996)
- *Local Monitoring*: This refers to the communication between components and there are two general methods for local monitoring known (Nagpal 2003, Sterrit and Hinchey 2006):
    - *Heart-beat monitors* are used to keep track of alive-signals which are sent by components to a parent or neighbors in regular intervals. They can detect the "death" or e.g. communication problems of the components they monitor.
    - *Pulse monitors* are a variation of the above but their signals contain more information, because they are sent as reflexes to give notice of an unwanted situation or event.
    - A similar, yet reversed mechanism is "apoptosis" ("dropping off"). It is a security mechanism that ensures a component self-destructs unless it receives a regular stay-alive-signal from its parent or neighbors.
- *Checkpoints and Consensus*: To coordinate parallel activities, all components that need to perform or finish an action before the others can proceed send out a halting signal. As long as the halting signal is present the next phase of the activity cannot be initiated by the components and thus ensures synchronization. (Nagpal 2003)
- *Differentiation and Context-Based Roles*: The role of each component is not necessarily pre-defined when the component is created and can also be of "general-purpose". Thus, initially coarse structures of general-purpose components organize themselves by the differentiation of their parts, which for example could depend on their location, their relation to other, neighboring components, or on the point in time. (Nagpal 2003)

- *Templates*: A template is here to be understood as a pattern that is used to construct another pattern. It is a structure existing in the IS which constrains and therefore guides the aggregation of components to form higher-level compositions. (Bonabeau et al. 1999, Camazine et al. 2003)
- *Stigmergy*: This is a mechanism for indirect communication between components, as information is gathered from work already finished or work in progress, i.e. an artificial construct. There are two types of stigmergy:
    - *Quantitative/ continuous*: The quantity of a stimulus is the important aspect of the communication. For example, the higher a value is, the more likely a component's reaction is.
    - *Qualitative/ discrete*: Stimuli are divided into different classes and the existence of a certain stimulus triggers specific actions of a component. This can lead to a chain of events if the triggered action itself is seen as a stimulus by some other component and causes an action and so on.

  Problems of stigmergy lie in its difficulties to control parallel activities and how to model an end-point for all activities that were initiated. (Camazine et al. 2003, Bonabeau et al. 1999, Di Marzo et al. 2004)

As stated above, these mechanisms should inspire adaptations and implementations to realize SO for IS and thereby reduce their complexity. Because implementations of artificial systems are not restricted by biological laws that might originally apply to the above mechanisms, combinations and hybrid approaches are possible and allow one mechanism to outweigh an other's shortcomings, for example "Checkpoints and Consensus" to parallelize "Stigmergy".

Alternatives

A discussion of SO could not hope to be complete if it left out the alternatives of SO. The three main alternatives are the following, according to Camazine et al. (2003):

- The *well-informed leader* is a single instance that monitors and controls all activities in the system. As the size of the system and the number of its components increase, crucial tasks of the leader such as acquiring information, processing this information, and communication with the components becomes more and more difficult and expensive.
- A system *blueprint* is a compact representation of the system's spatial and temporal structure. Depending on the size of the system such a blueprint can be very expensive to encode (in terms of space needed) and while it defines "*what"* is to be built, the "*how"* is left out.
- A *recipe* adds the "how" to the "what" of a blueprint and provides sequential instructions to be performed for building and maintaining a system. However, while such a detailed plan is feasible for single executing components, it becomes rather difficult for groups due to synchronization and collaboration issues.

As we can see above, the alternatives of SO are already quite common in IS, be it the administrator as the (hopefully) well-informed leader or the system's formal architecture description as a blueprint. SO for IS can be seen as a chance of support for these already used techniques, i.e. allowing administrators to focus on major tasks instead of simple and tedious ones.

CONCLUSIONS AND OUTLOOK

Modern information systems face the problem of increasing complexity, e.g. due to the adaptability requirements of competitive, global markets. In this paper, we discussed the general idea of self-organization for IS, i.e. IS using SO mechanisms to transform themselves into new and improved configurations. While general concepts and mechanisms were discussed to provide ideas for further research in this area, the two greatest challenges for SO in IS remain still unsolved:

- The initial effort to integrate SO capabilities and adaptations of successful mechanisms into IS is high.
- Emergent behavior of SO is hard to control and even harder to predict.

To address these issues and to exploit the potential benefits of SO for IS, we are working on integrating SO capabilities and mechanisms into business-relevant application scenarios. The focus of our current research is on the integration of self-organization mechanisms into SOA (Papazoglou, Traverso, Dustdar, Leymann, and Krämer 2006), building scalable and dependable distributed service ecosystems.

As a first step towards self-organizing SOA, we developed a distributed monitoring and deviation handling architecture for service-based workflows named Automated Monitoring & Alignment of Services (AMAS.KOM), allowing decentralized monitoring of service execution as well as the self-sustained handling of service level violations (Repp, Eckert, Schulte, Niemann, Berbner, and Steinmetz 2008). As a next step, we will extend AMAS.KOM towards being a test bed for different SO mechanisms. This will allow us to get an improved understanding of how SO works in actual service-oriented scenarios, how it can be controlled, and how its performance can be measured.

Using test bed experiments and simulation technology is necessary, because unlike in nature, enterprises and other users of IS cannot fall back to trial and error approaches such as natural selection which improve and adapt complex systems over billions of years.

ACKNOWLEDGMENTS

This work is supported in part by E-Finance Lab e.V., Frankfurt am Main, Germany and BearingPoint Management and Technology Consultants.

REFERENCES

1. Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999) Swarm Intelligence: From Natural to Artificial Systems (Santa Fe Institute Studies in the Sciences of Complexity Proceedings), Oxford University Press, USA.
2. Brooks, F. P. (1995) The Mythical Man-Month: Essays on Software Engineering, 20th Anniversary Edition, Addison-Wesley Professional.
3. Camazine, S., Deneubourg, J., Franks, N. R., Sneyd, J., Theraulaz, G., and Bonabeau, E. (2003) Self-Organization in Biological Systems: (Princeton Studies in Complexity), Princeton University Press.
4. Capra, F. (1997) The Web of Life: A New Scientific Understanding of Living Systems, Anchor.
5. Di Marzo, G., Foukia, N., Hassas, S., Karageorgos, A., Mostéfaoui, S. K., Rana, O. F., Ulieru, M., Valckenaers, P., and Van Aart, C. (2004) Self-Organisation: Paradigms and Applications, Lecture Notes in Computer Science: Engineering Self-Organising Systems, Springer, pp. 1-19.
6. Ganek, A.G. and Corbi, T. A. (2003) The Dawning of the Autonomic Computing Era. IBM Systems Journal (Autonomic Computing), 42(1):5–18.
7. Goldstein, J. (1999) Emergence as a Construct: History and Issues, Emergence **1**(1), 49-72.
8. Hewlett-Packard (2008) Adaptive Infrastructure, http://www.hp.com/go/ai
9. Hinchey, M. G. and Sterritt, R. (2006) Self-Managing Software, Computer **39**(2), 107.
10. Hinchey, M. G. and Sterritt, R. (2007) 99% (Biological) Inspiration..., in *EASE '07: Proceedings of the Fourth IEEE International Workshop on Engineering of Autonomic and Autonomous Systems*, IEEE Computer Society, Washington, DC, USA, pp. 187-195.
11. Hinchey, M. G., Sterritt, R., and Rouff, C. (2007) Swarms and Swarm Intelligence, Computer **40**(4), 111-113.
12. Holland, J. H. (1996) Hidden Order: How Adaptation Builds Complexity (Helix Books), Addison Wesley Publishing Company.
13. International Business Machines Corp. (IBM) (2008) Autonomic Computing, http://www.ibm.com/autonomic/
14. International Business Machines Corp. (IBM) (2006) An Architectural Blueprint for Autonomic Computing, http://www-03.ibm.com/autonomic/pdfs/AC_Blueprint_White_Paper_4th.pdf.
15. Jacob, Bart, Lanyon-Hogg, Richard, Nadgir, Devaprasad K, and Amr F Yassin (2004) A Practical Guide to the IBM Autonomic Computing Toolkit, IBM Redbooks, http://www.redbooks.ibm.com/redbooks/pdfs/sg246635.pdf.
16. Jann, J., Browning, L. M., and Burugula, R. S. (2003) Dynamic Reconfiguration: Basic Building Blocks for Autonomic Computing on IBM pSeries Servers, IBM Systems Journal (Autonomic Computing) **42**(1), 29-37.
17. Kauffman, S. (1996) At Home in the Universe: The Search for the Laws of Self-Organization and Complexity, Oxford University Press, USA.
18. Kephart, J. O. and Chess, D. M. (2003) The Vision of Autonomic Computing, IEEE Computer **36**(1), 41-50.
19. Krafzig, D., Banke, K., and Slama, D. (2004) Enterprise SOA: Service-Oriented Architecture Best Practices (The Coad Series), Prentice Hall PTR.
20. Liu, J. and Tsui, K. C. (2006) Toward Nature-Nnspired Computing, Commun. ACM **49**(10), 59-64.
21. Melcher, B. and Mitchell, B. (2004) Towards an Autonomic Framework: Self-Configuring Network Services and Developing Autonomic Applications, Intel Technology Journal **8**(4), 279-290.
22. Microsoft Corp. (2008) Dynamic Systems Initiative, http://www.microsoft.com/business/dsi

23. Nagpal, R. (2003) A Catalog of Biologically-Inspired Primitives for Engineering Self-Organization, in *Engineering Self-Organising Systems*, pp. 53-62.
24. Newcomer, E. and Lomow, G. (2004) Understanding SOA with Web Services (Independent Technology Guides), Addison-Wesley Professional.
25. Papazoglou, M. P. (2003) Service-Oriented Computing: Concepts, Characteristics and Directions, 4th International Conference on Web Information Systems Engineering (WISE 2003), IEEE, Rome, Italy, pp. 3-12.
26. Papazoglou, M. P., Traverso, P., Dustdar, S., Leymann, F., and Krämer, B. J. (2006) Service-Oriented Computing Research Roadmap, in *Dagstuhl Seminar Proceedings on Service Oriented Computing (SOC).*
27. Repp, N., Eckert, J., Schulte, S., Niemann, M., Berbner, R., Steinmetz, R (2008) Towards Automated Monitoring and Alignment of Service-based Workflows, in *IEEE International Conference on Digital Ecosystems and Technologies 2008 (IEEE DEST 2008).*
28. Shalizi, C. R. (2001) Causal Architecture, Complexity and Self-Organization in Time Series and Cellular Automata, PhD thesis, The University of Wisconsin, Madison.
29. Sommerville, I. (2006) Software Engineering, Addison Wesley.
30. Sterritt, R. and Hinchey, M. G. (2005) Autonomicity -- An Antidote for Complexity?, in *CSBW '05: Proceedings of the 2005 IEEE Computational Systems Bioinformatics Conference - Workshops (CSBW'05)*, IEEE Computer Society, Washington, DC, USA, pp. 283-291.
31. Sterritt, R. and Hinchey, M. G. (2006) Biologically-Inspired Concepts for Self-Management of Complexity, in *ICECCS '06: Proceedings of the 11th IEEE International Conference on Engineering of Complex Computer Systems*, IEEE Computer Society, Washington, DC, USA, pp. 163-168.