Toward the Design of Self–Organizing Dynamics

Jan Sudeikat[1,2]

[1]Multimedia Systems Laboratory (MMLab),
Faculty of Engineering and Computer Science,
Hamburg University of Applied Sciences,
Berliner Tor 7, 20099 Hamburg, Germany
sudeikat@informatik.haw-hamburg.de
[2]Distributed Systems and Information Systems (VSIS),
Computer Science Department, University of Hamburg,
Vogt–Kölln–Str. 30, 22527 Hamburg, Germany
4sudeika@informatik.uni-hamburg.de

Abstract: A growing demand for distributed and decentralized software, together with an increasing inherent complexity of these systems challenges traditional software engineering practices. Recently, the utilization of self–organizing processes has been proposed as a way to relieve development efforts. While Agent–Oriented Software Engineering (AOSE) provides suitable abstractions and implementation frameworks for these system dynamics, the design of phenomena that rise solely from entity interactions contradicts traditional top–down development strategies. This paper describes work in progress and outlines a doctoral thesis that addresses the analysis and design of self–organizing distributed systems in Multi–Agent Systems (MAS), i. e. collections of autonomous, pro–active entities.

1 Introduction

Today's application domains are increasingly dynamic and distributed, demanding corresponding software architectures and decentralized coordination mechanisms. These domain properties as well as the continuously growing *essential complexity of software* challenge software engineering approaches to system analysis, design, and maintenance.

Agent Oriented Software Engineering (AOSE) [HSG05] is a prominent approach to the development of complicated distributed software systems. *Agents*, i. e. autonomous and pro–active entities are proposed as a basic design and development metaphor. Since highly dynamic and distributed application domains lend themselves to be understood as collections of dependent and collaborating actors, this metaphor provides appropriate abstractions. Applications result from agent interplay, and system dynamics have been reported to resemble *Complex Adaptive Systems* (CAS) [BY02]. The study of CAS examines collections of interconnected entities that exhibit globally coherent, adaptive system behaviors, due to cooperation and competition of individuals. Recognized examples are *insect colonies*, that establish shortest trails to resources and sort their nests, due to local agent coordination.

These emergent phenomena result from self–organizing processes that establish and maintain system configurations solely by distributed entity coordination, i.e. in the absence of central control units. Deliberate control of theses dynamics – observable in CAS as well as artificial systems [Mog05] – challenges engineering efforts but promises *robust*, *scalable* and *adaptive* computational systems and has therefore been identified as a strategic challenge to IT research[1]. The autonomous nature of agents and the complex interactions between them not only enable the purposeful creation of self–organizing processes, but can introduce them implicitly, leading to unintended system properties [Mog05].

Awareness is rising that CAS phenomena are inherently possible in MAS and poses the open challenge for AOSE research to provide means to handle these phenomena in development efforts (e. g. see [HSG05] page 4). Current engineering approaches to MAS can be distinguished between top–down development *methodologies* and bottom–up, *experimentation*–based procedures. The former ones aim at generic agent–based applications, although they need to support the detection and prevention of possible occurrences of self–organizing phenomena; while the latter ones intend particularly these phenomena and would benefit from dedicated modeling techniques, design principles and validation approaches. The here outlined work examines how both approaches can be combined, by providing a novel modeling approach, to be supported by tailored analysis and design processes as well as corresponding tool support.

2 Background and Related Work

Academic and industrial development efforts provide a number of sophisticated agent platforms and agent–based programing languages [BPL06]. Building on these efforts, *Agent Oriented Software Engineering* (AOSE) issues gain in importance for MAS research. Top-down development procedures for MAS are supported by *agent–oriented* design approaches that are typically biased in favor of certain agent architectures and organizational approaches that define organizational structures and interaction networks in terms of *roles* and *groups*. These approaches are opposed by bottom–up *emergentist* procedures that particularly focus on self–organizing dynamics.

Designers have specific system–wide, *macroscopic* properties in mind – derived from system requirements – when structuring the future software system. In the case of self–organizing systems the inherent non–linearity of agent interactions impair straightforward predictions of system dynamics. Therefore, top–down development methodologies provide sets of sophisticated design tools, typically ignoring the possibility of self–organizing dynamics (see e.g. [HSG05]), while emergentist approaches rely on experimentation procedures, commonly focusing on *reconstruction* of well–examined phenomena established by cataloged coordination mechanisms [SGK06]. Therefore, development teams usually (1) identify a well understood coordination mechanism and (2) map its constituent parts to the actual application domain. Finally the (3) behavior of prototype implementations is examined and parameters – controlling agent/environment properties – are adjusted [SR06].

[1]e. g. by the *Feldafinger Kreis*: http://www.feldafinger-kreis.de/ergebnisse/spezi_inhalt_01.htm

3 Toward a Multi–Level Modeling Approach to MAS

Inspired by established multi–level descriptions for CAS behavior analysis, we propose intermediate – *mesoscopic* – description levels to mediate between top–down and bottom–up development efforts. This modeling notion allows to relate system behaviors to averaged individual agent behaviors, facilitating bottom–up analysis of implementations [RS06] and top–down (re-)design via predictive system simulations [SR06]. *Macroscopic* system models illustrate possible system configurations. Denoting the space of possible system states aids to draw a distinction between *desired* structures and/or emergents from *unintended* ones, facilitating developers to identify the intended transitions between them that should be enforced by the self–organizing dynamics [SR06]. *Microscopic* modeling levels guide agent implementation, following specific agent architectures.

Transitions between macroscopic configurations can be explained by *mesoscopic* models which introduce intermediate artifacts, abstracting from microscopic agent behaviors (cf. figure 1) [RS06]. Abstraction is done by averaging out multiple microscopic agent behaviors and replacing them by a few abstract states. Appropriate abstraction is a considerable modeling effort, which requires classifying agent states according to their contribution to the macroscopic dynamics. It has been found that these models are useful to mathematically describeing these processes that result from distributed coordination mechanisms in MAS [RS06]. The introduced mesoscopic states resemble *roles* which agents play in respect of the exhibited macroscopic system dynamics.

Current work addresses support for the application development life-cycle – from requirements analysis via MAS design and predictive simulations to system validation – of self–organizing MAS via mesoscopic modeling. Open issues concern how intended system behaviors can be analyzed and transferred to appropriate system abstractions that guide MAS implementation. As the *model driven development* of applications via mesoscopic models is currently impaired by the diversity of applied coordination mechanisms and corresponding agent platforms, an appropriate reference agent model is under development. Mesoscopic modeling will be facilitated by a distributed analysis/testbed infrastructure, allowing time series examination and their (semi-)automated comparison to expected system behaviors, transferring MAS simulation to behavior space testing.

Circumstances and Procedure

Careful supervision by Prof. Dr. Renz at HAW Hamburg initiated research how theories of complex system dynamics effect software engineering for self–organizing systems. Research is based on an inspiring cooperation with Prof. Dr. Lamersdorf from Hamburg University, combining theory and engineering practices. After post graduate studies in theoretical computer science the PhD proposal has been accepted in summer 2006 and the completion of the thesis is scheduled for three subsequent years, where the above discussed challenges are to be addressed in an iterative process. In three distinct steps the necessary analysis support and modeling notions will be revised, alternated by the treatment (modeling, implementation and evaluation) of appropriate cases studies. The resulting insights will trigger incremental development of appropriate testbed and validation facilities.

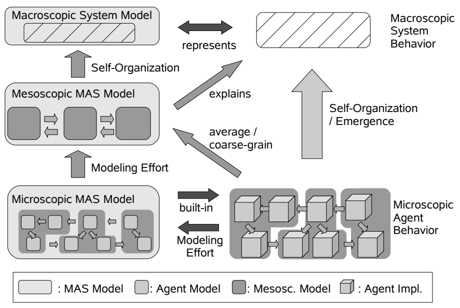

Figure 1: The relation between agent implementation behaviors (right) and modeling (left). Mesoscopic models average out agent behaviors and explain macroscopic structures [SR06].

4 Conclusions

In this paper, the challenges to the development of self–organizing dynamics in software have been outlined. To support *engineering* efforts, a novel multi–level modeling approach has been presented, facilitating system analysis and redesign. Future work will address three areas of examination: First, *theoretical* issues concern the utilization of formal tools to express and examine mesoscopic models; Secondly, *Software–Engineering* issues arise, demanding an examination of how this modeling notion can be embedded in current AOSE best practices; Finally, *practical* issues concern suitable MAS analysis tool support.

References

[BPL06] Lars Braubach, Alexander Pokahr, and Winfried Lamersdorf. Tools and Standards. In *Multiagent Engineering - Theory and Applications in Enterprises*, Springer Series: International Handbooks on Information Systems, 3 2006.

[BY02] Y. Bar-Yam. General Features of Complex Systems. In *Encyclopedia of Life Support Systems (EOLSS)*, Oxford ,UK, 2002. EOLSS Publishers.

[HSG05] Brian Henderson-Sellers and Paolo Giorgini, editors. *Agent-oriented Methodologies*. Idea Group Publishing, 2005. ISBN: 1591405815.

[Mog05] Jeffrey C. Mogul. Emergent (Mis)behavior vs. Complex Software Systems. Technical Report HPL-2006-2, HP Laboratories Palo Alto, 2005.

[RS06] Wolfgang Renz and Jan Sudeikat. Emergent Roles in Multi Agent Systems - A Case Study in Minority Games. *KI - Zeitschrift fuer Kuenstliche Intelligenz*, 2006.

[SGK06] G. D. M. Serugendo, M. P. Gleizes, and A. Karageorgos. Self–Organisation and Emergence in MAS: An Overview. In *Informatica*, volume 30, pages 45–54, 2006.

[SR06] Jan Sudeikat and Wolfgang Renz. On the Redesign of Self–Organizing Multi–Agent Systems. *International Transactions on Systems Science and Applications*, 2(1):81–89, 2006.