1
Introduction

Designing decentralized and distributed applications in an environment which is dynamic, noisy and unpredictable requires an approach that is robust, flexible, adaptive and scalable. Software architects are increasingly relying on self-organizing mechanisms to design distributed systems with these features. Self-organizing mechanisms are mainly bio-inspired mechanisms, such as ants foraging or morphogen gradients [Serugendo 2006], or socially-inspired, such as market-based [Gibney et al. 1999]. Or they can be non naturally-inspired, which are mechanisms based on the self-properties [Kephart 2003], tag-based [Hales 2006], or trust-based [Chatterjee 2009].

In a self-organizing solution, each agent acquires and maintains information about its environment and neighbors without external control. The emergent behavior may evolve or change over time [Serugendo 2006, De Wolf 2007]. The environment is structurally distributed; in other words, at any point in time, no centralized entity has complete knowledge of the state of the environment as a whole.

In this context, there are two tracks that drive the research in this thesis. First, it is often far from obvious how the individual agent behavior need to be designed to meet the overall design goal. As it is the case with any new software engineering paradigm, the successful and widespread deployment of self-organizing systems require: notations that explore the use of self-organizing related abstractions and promote the traceability from the design models to code, and engineering methods that provides know-how and guides an engineer during an application design. As the overall goal of a decentralized systems designed with self-organization is novel with regard to the agents, the validation of the system outcome is an essential and hard task.

Second, there is a large nature of applications that can be designed (or redesigned) as a self-organizing systems. The different approaches can be divided depending on the mechanisms [Serugendo 2006], for instance, direct interactions between agents using basic principles such as broadcast and localization; or indirect interactions between agents and stigmergy. Furthermore, a designer may decide to model environments using various underlying structures. For ex-

ample, an environment can be modeled as a graph, a grid, a continuous space or a combination of these. On the other hand, all of this different applications share a common behavior inherited from self-organizing mechanisms. Therefore, the second research track is to promote software reuse in self-organizing systems. Software reuse [Krueger 1992] has been a lofty goal for Software Engineering (SE) research and practice, as a means to reduce development costs and improve quality. To increase chances of achieving useful reuse, we need to establish tools and methods needed for reuse and design for reuse.

Results from both research tracks are complementary and the interdisciplinary research of this thesis addresses the opportunity to bridge the gap between notational support in the design phase and a simulation-based self-organizing architecture followed by an engineering and autonomic validation method that promotes reuse. The presence of design rationale is important for knowing whether some design decisions need to be reconsidered when an architecture is used in a different context.

1.1 Problem Statement

While there is plenty of literature about the analysis of self-organizing and emergent mechanisms, the interest in engineering aspects grew only recently. Van Parunak and Bruckner first proposed a design guide for swarming systems engineering based on principles [Parunak 2004]. De Wolf [De Wolf 2007] defined a full life-cycle methodology based on the Unified Process customized to explicitly focus on engineering macroscopic behavior of such kind of systems. Luca Gardelli [Gardelli 2008] proposed a meta-model and a methodological approach for engineering self-organizing multi-agent systems. And finally, ADELFE provides a methodology to support self-organization [Bernon 2004], more precisely for the Adaptive MAS theory (AMAS). De Wolf and Gardelli also proposed architectural design patterns for designing self-organizing mechanisms.

In the self-organizing community, there is a common belief that to build self-organizing systems computer scientists must think flows rather than transitions, and to ensure that these flows include closed loops [Parunak 2004, De Wolf 2007]. To this end De Wolf proposed the use of UML 2 Activity Diagrams. Gardelli didn't propose any notation to design information flows, neither ADELFE. Although information flows are essentials to support self-organization, an important matter is to design considering modularity and reuse. In addition, modularity should be easy to work with because modules can be easily understood in isolation, and changes or extensions to functionality would be easily localized. And, design information flows using UML 2 Activity

Diagrams as it was proposed by De Wolf does not help on modularity and reuse of models. Furthermore, in the validation process, the main issue consists of evaluating the capacity to reach an organization that is able to fulfill the goal of the system as a whole once the system is started, i.e, the organization converges to the desired guarantees or macroscopic properties [De Wolf 2007]. Hence we need modeling approach that combines information flows, state dynamics and software engineering principles as reuse and modularity.

To understand the problem of an architecture approach to self-organizing systems, let's take a look into three different problems which are the case studies of this thesis:

– **The Automated Guided Vehicles**

An AGV transportation system uses multiple computer-guided vehicles in warehouses. Each vehicle, or AGV, moves loads (e.g. packets, materials) in a warehouse from the pickup to the drop stations and can only perform a limited set of local actions, such as move, pick up load, and drop load. The goal is to efficiently transport incoming loads to their destination.

– **Contracts that Govern Emergent Multi-Agent Systems**

Governing the behavior of autonomous agents in multi-agent systems to reach overall system benefit has long been an active area of research. One approach of recent prevalence is to provide agents with explicit specifications of what they should, should not or may do within the system, i.e. normative statements or norms. In a business setting, these norms exactly mirror the contractual agreements made between business organizations. In this problem, a simulation technique designed for investigating and tuning emergent behavior in multi-agent systems can be combined with an approach to modeling norms of the complexity found in business contracts. Using an aerospace case study, we need to aid in the refinement of such contracts by exposing the consequences of contract variations.

– **Computational Modeling of Stem Cell Behavior**

A stem cell is a primitive cell that can either self-renew (reproduce itself) or give rise to more specialized cell types [Loeffler 2003, Lord 1997]. Nowadays, stem cells are cultivated in the lab in order to differentiate into a specific mature cell [Rehen 2006]. Today it is hard to predict the stem cell behavior under some substances. Moreover, the entire infrastructure

necessary to maintain a stem cell culture is very expensive and many stem cells are wasted if the injected substance does not lead the culture to the desired mature cell. Thus, stem cell simulation is a powerful tool for reducing such costs and accelerating the stem cell therapy process. At the agent-based cell computational model, the environment is a distributed entity composed of reactive and pro-active agents. The cells can perceive and signal the environment. In this problem, we need to maximize the number of neuron progenitor cells, and minimize cell deaths or overpopulation.

In the AGV case study, the decentralized control is achieved through the use of Gradient Fields [Mamei 2004, De Wolf 2007a] that indirectly guide the automated vehicles. It enables aggregation and calculation of extra information while flowing through the warehouses. In the Contracts' problem, it is not easy to specify norms such that the desired emergent macroscopic behavior is achieved. The self-organization is characterized by the time the aircraft is flying in combination with the set of active norms. And finally, in the stem cell problem, the stem cell behavior is a self-organizing system by nature and we need to find the best configuration to achieve the desired emergent behavior. The three problems share the same architecture, but no existent architecture is able to fulfill all the requirements desired for them all. The entities share the same behavior as local perception, coordination mechanism, distributed environment management and could seize the same architecture.

Moreover, an architecture that provides the underline engineering and validation process, can help the development of not only these problems, but of any problem that share the same dynamics.

1.2 Goals

This thesis has as its main goal to improve the state of art of the engineering of self-organizing emergent multi-agent systems. To this end it aims to provide a design method composed of a notation and architecture, and an engineering method. Based on the problem statement, to achieve these tasks, this thesis has the following subgoals:

– Provide a notational support based on UML 2 [OMG 2005]. This notational support comprises a customization of the UML 2 meta-model and its main goal is to support the design of information flows combined with state dynamics and software engineering principles as reuse and modularity;

- Investigate the applicability of the proposed notation in the modeling of self-organizing architectural patterns;
- Provide a simulation-based architecture for self-organizing emergent multi-agent systems. More specifically, this architecture must:
  - Provide simulation features;
  - Provide core components underlying the environment, including different structures and allow creativity in the design of self-organizing multi-agent systems;
  - Provide coordination mechanisms to support reuse of self-organizing mechanisms;
  - Provide mechanisms for action selection to an integral model that includes support for reusable autonomic validation mechanisms.
  - Provide validation mechanisms to support the micro-macro relationship understanding.
- Provide simulation-based engineering guidelines to support the application of the notational support and the architecture instantiation;
- Evaluate the architecture through three different application domains: the automated guided vehicles, the contracts that govern emergent multi-agent systems, and the stem cell computational model.

1.3 Contributions

The main contributions of this thesis address the following areas: Agent-Oriented Software Engineering, Self-Organizing Software Engineering and Autonomic Computing.

From the Agent-Oriented Software Engineering point of view, the main contributions are the notational support, with the UML customizations, and the architectural validation features. Self-organizing agents need to be coordinated in some way, and those customizations help with this task. Also, emergent behavior is an inherent multi-agent system characteristic, and the architectural validation features help with the monitoring and controlling of agents misbehavior.

From the Self-Organizing Software Engineering point of view, the main contribution is the application of an agent-oriented technology to build those systems that provides design support, engineering guidelines, an architecture and support to good practices such as reuse and modularity. The architecture provides an asset base engineers can draw from when developing self-organizing applications.

From the Autonomic Computing point of view, the main contribution for this area consists of the architectural validation features and the architecture itself, since the emergent misbehavior can be avoided with these technologies.

The application of the engineering guidelines and simulation-based self-organizing architecture in three complex industrial application are used to validate the above mentioned results. We have applied the various mechanisms for architectural design of self-organizing systems to meet the functionality and satisfy the desired guarantees of the system. The insights derived from the architectural design of these applications have considerably contributed to the development of the proposed architecture.

Finally, two other contributions were achieved with the case studies. The first one is the proposal of an original approach to the problem of designing contracts which produce a beneficial emergent macroscopic behavior through a systematic simulation approach using norm specifications. We have showed, using an aerospace case study, that our approach can also aid in the refinement of such contracts by exposing the consequences of contract variations.

The second is the computational modeling use of biological system. There is an intensive research attempting to provide computational methods to the modeling and simulation of biological systems, such as differential equations and cellular automata. However, they fail from the usability point of view to express the models, their processes, dynamic environment and partial view as already mentioned by [d'Inverno 2005]. We illustrate through the stem cell computational modeling case study how the proposed technologies can help with this task.

1.4 Structure of the Thesis

Here we provide an overview of the thesis structure and summaries for each chapter. The original contributions of this thesis are mainly concentrated in Chapters 3, 4 and 5.

Chapter 2 Background This chapter provides some necessary background information to allow the understanding of the thesis. In particular, we discuss definitions of both Agent-based Simulation, Self-Organization and Emergence. It further describes the three case studies used in this thesis in detail, and the specific problems that must be solved through our approach.

Chapter 3 Notational Support Here we provide UML 2 customizations that support the modeling of self-organizing systems. It includes a metamodel that provides foundations for the models and statecharts extensions that help modeling coordination and information flows. We introduce a pat-

tern language built by considering standard self-organizing design patterns as an instance of an exemplar application of the notational support utilization.

Chapter 4 SSOA: A Simulation-based Self-Organizing Architecture In this chapter we describe the requirements for a simulation-based self-organizing architecture, the simulation-based engineering method that should be applied when using the architecture, the architecture meta-model with its all entities, relationships, autonomic validation components, life cycle and dynamics. Then we introduce the framework that implements the architecture [Gatti 2009] and that was instantiated for the case studies.

Chapter 5 Case Studies This chapter discusses the case studies that have been used in this thesis. Although the stem cell computational modeling has been the first case study tackled and the one that motivated the whole work developed in this thesis, it is the last case study presented due to its complexity and scalability problem[1].

Chapter 6 Related Work We present the state of the art and the differences between this thesis and the works related to the modeling and designing of self-organizing systems, including methodologies, architectures for self-organizing systems, validation and verification methods.

Chapter 7 Conclusions and Future Work We conclude by summarizing the thesis, by highlighting the contributions, and limitations, and by proposing future works.

[1]The solution for this case study had three versions: a 2D low performance version, designed using MAS-ML [Silva 2007] and with no underline self-organizing architecture [Gatti 2007a], a 2D version designed using AUML [Bauer 2001] and built on top of the MASON toolkit [Luke 2004] but with no self-organizing architecture [Gatti 2008], later we produced the first version of the architecture presented in Chapter 4 with a 3D continuous floating-point version [Faustino et al. 2008]. After this, we have evolved the architecture followed by the development of the other two case studies: the automated guided vehicles, and the contracts that govern multi-agent systems. We finally refactored the stem cell case study in order to be able to use all the common architecture features and those last results are described for each case study in this thesis.