Foreword

The increasing cooperation and convergence of various kinds of computing entities, i.e., computers, cellphones, personal digital assistants, house appliances, etc., is fundamentally changing the way we view computers and software. The size, increasing complexity and the great potential of future applications for change (e.g., community support, collaborative work and supervision) make centralized and direct control by the programmer nearly impossible. It is thus natural to delegate more autonomy and initiative to various software modules and to provide them with cooperation abilities. Multi-agent systems have been proposed as a conceptual framework to help design and construct such large-scale autonomous and cooperative computing systems.

We can analyze multi-agent systems as a programming paradigm in terms of the evolution of programming in general. We may for instance observe three dimensions of progress: (1) higher levels of abstraction for entities processed or/and exchanged – from bits, and then objects and messages, to agents, intentions and plans; (2) later binding times, i.e., deferring the decision regarding what actual code is to be executed – from procedure call, and then method invocation, to action selection by an autonomous agent; and (3) more flexible coupling between software modules – from procedures, and then objects, components and events, to knowledge-based organizations of agents. Multi-agent systems may thus be seen as an integral part of the evolution of programming.

There is currently a growing body of experience on how to construct multi-agent architectures and platforms based on more conventional technology, objects, Java and components. Interoperability has recently been an important concern, notably through the FIPA Agent Communication Language standard, which builds on object interoperability standards (such as CORBA) and extends their levels of abstraction (e.g., expliciting interaction patterns and protocols, as well as ontologies of knowledge, within a communication). We also see now various attempts and studies for using novel approaches and technologies such as Model Driven Architecture (MDA) and Aspect-Oriented Programming (AOP) in the design and construction of multi-agent systems. Last, the inherent decentralization and autonomy of multi-agent systems also raise new questions about dependability and security.

From the software engineering perspective, we believe the challenges are perhaps even bigger. The decentralized, autonomous and adaptive nature of agents, combined with the large scale, mobility and general dynamicity of their environment support (e.g., ad hoc networks, with associated security and robustness concerns), make it difficult to rely on traditional assumptions of software predictability. In other words, the traditional “defensive/pessimistic” approach – statically safeguarding as much as possible the behavior of the program (through specifications, types, assertions, etc.) – reaches its limits and should be complemented (and not replaced!) by a “proactive/optimistic”

approach, providing the agents with abilities to adapt to unexpected individual and collective behaviors. Another concern is to include also the users, as agents, from the initial stages of the design. Indeed, our ultimate goal is to provide a symbiotic collaboration between artificial agents and human agents, as opposed to confining users, either as supervisers with explicit control or as end-users with little initiative.

For designing practical methodologies, there is still a debate in the community on whether agent-oriented methodologies should be affiliated to current (object-oriented) methodologies or should be deeply restated (for instance by focusing analysis on social concepts such as roles and organization rather than on individual objects or agents). In any case, future methodologies will still need steps (such as analysis, design, modeling, measurements, etc.) as well as related techniques (such as requirements analysis, meta-modeling, notations and metrics). As a consequence, there is currently much activity in studying how such steps or techniques may be partly reused from current technology, adapted or completely rethought.

As a conclusion, in order to achieve these challenges, we need to organize a research community at the crossing of software engineering, programming and multi-agent systems, with a concern for scalability of solutions. This book, the third volume of the very good series on "Software Engineering for Large-Scale Multi-agent Systems," includes several important studies and proposals along the lines of the various perspectives we just sketched, and thus represents a very good contribution to that research agenda.

Jean-Pierre Briot
Paris, December 2004

Preface

Advances in networking technology in the last few years have turned agent technologies into a promising paradigm for engineering complex distributed software systems. So far they have been applied to a wide range of application domains, including e-commerce, human-computer interfaces, telecommunications, and concurrent engineering. Multi-agent systems (MASs) and their underlying theories provide a more natural support for ensuring important properties, such as autonomy, mobility, environment heterogeneity, organization and openness. Nevertheless, a software agent is an inherently more complex abstraction, posing new challenges for software engineering. Without adequate development techniques and methods, MASs will not be sufficiently dependable, trustworthy and extensible, thus making their wide adoption by the industry more difficult.

Large MASs are complex in many ways. When a set of agents interact over heterogeneous environments, several problems emerge. This makes their coordination and management more difficult and increases the probability of exceptional situations, security holes and unexpected global effects. Moreover, as users and software engineers delegate more autonomy to their MASs and put more trust in their results, new concerns arise in real-life applications. Yet many of the existing agent-oriented solutions are far from ideal; in practice, systems are often built in an ad hoc manner, are error-prone, not scalable, not dynamic, and not generally applicable to large-scale environments. If agent-based applications are to be successful software engineering approaches will be needed to enable effective scalable deployment.

The papers selected for this volume present advances in software engineering approaches to the development of realistic multi-agent systems, demonstrating a broad range of techniques and methods used to cope with the complexity of systems like these and to facilitate the construction of high-quality MASs. Furthermore, the power of agent-based software engineering is illustrated using examples that are representative of real-world applications. These papers describe experiences and techniques associated with large MASs in a variety of problem domains.

A comprehensive selection of case studies and software engineering solutions for MASs applications, this book provides a valuable resource for a vast audience of readers. The main target readers for this book are researchers and practitioners who want to keep up with the progress of software engineering in MASs, individuals keen to understand the interplay between agents and objects in software development, and those interested in experimental results from MAS applications. Software engineers involved with particular aspects of MASs as part of their work may find it interesting to learn about using software engineering approaches in building real systems. A number of chapters in the book discuss the development of MASs from requirements and architecture specifications to implementation.

One key contribution of this volume is the description of the latest approaches to reasoning about complex MASs.

This book brings together a collection of 16 papers addressing a wide range of issues in software engineering for MASs, reflecting the importance of agent properties in today's software systems. The papers presented describe recent developments in specific issues and practical experience. The research issues addressed include (i) integration of agent abstractions with other software engineering abstractions and techniques (such as objects, roles, components, aspects and patterns), (ii) specification and modelling approaches, (iii) innovative approaches for security and robustness, (iv) MAS frameworks, and (v) approaches to ensuring quality attributes for large-scale MASs, such as dependability, scalability, reusability, maintainability and adaptability. At the end of each chapter, the reader will find a list of interesting references for further reading. The book is organized into four parts, which deal with topics related to (i) Agent Methodologies and Processes, (ii) Requirements Engineering and Software Architectures, (iii) Modelling Languages, and (iv) Dependability and Coordination.

This book is a continuation of two previous volumes[1,2]. The main motivation for producing this book was the *3rd International Workshop on Software Engineering for Large-Scale Multi-agent Systems* (SELMAS 2004)[3], organized in association with the 26th International Conference on Software Engineering, held in Edinburgh, UK, in May 2004. SELMAS 2004 was our attempt to bring together software engineering practitioners and researchers to discuss the multifaceted issues arising when MASs are used to engineer complex systems. It was later decided to extend the workshop scope, inviting several of the workshop participants to write chapters for this book based on their original position papers, as well as other leading researchers in the area to prepare additional chapters. Following an extensive reviewing process involving more than 40 reviewers, we selected the papers that appear in this volume.

We are confident that this book will be of considerable use to the software engineering community by providing many original and distinct views on such an important interdisciplinary topic, and by contributing to a better understanding and cross-fertilization among individuals in this research area. It is only natural that the choice of contributors to this book reflects the personal views of the book editors. We believe that, despite the volume of papers and work on software engineering for MASs, there are still many interesting challenges to be explored. The contributions to this book are only the beginning. Our thanks go to all our au-

---

[1] Garcia, A., Lucena, C., Castro, J., Zambonelli, F., Omicini, A. (eds.): Software Engineering for Large-Scale Multi-agent Systems. Lecture Notes in Computer Science, vol. 2603, Springer, April 2003.

[2] Lucena, C., Garcia, A., Romanovsky, A., Castro, J., Alencar, P. (eds.): Software Engineering for Multi-agent Systems II. Lecture Notes in Computer Science, vol. 2940, Springer, February 2004.

[3] Choren, R. el al.: Software Engineering for Large-Scale Multi-agent Systems – SELMAS 2004 (Workshop Report). ACM Software Engineering Notes, Vol. 29, N°. 5, September 2004.

thors, whose work made this book possible. Many of them also helped during the reviewing process. We would like to express our gratitude to Alfred Hofmann from Springer for recognizing the importance of publishing this book. We also acknowledge the support and cooperation of Anna Kramer and Judith Freudenberger, who helped us in the preparation of this volume. In addition, we would like to thank the members of the Evaluation and Program Committee who were generous with their time and effort when reviewing the submitted papers.

December 2004

Ricardo Choren
Alessandro Garcia
Carlos Lucena
Alexander Romanovsky