The Journal of Systems and Software 86 (2013) 890–904

Contents lists available at SciVerse ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Contributions to the emergence and consolidation of Agent-oriented Software Engineering

Carlos Lucena[a,*], Ingrid Nunes[a,b]

[a] Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Brazil
[b] King's College London, Strand, London WC2R 2LS, United Kingdom

ARTICLE INFO

Article history:
Received 20 January 2012
Received in revised form 12 May 2012
Accepted 7 September 2012
Available online 14 September 2012

Keywords:
Multiagent systems
Agent-oriented Software Engineering
LES
PUC-Rio
SBES 25 years

ABSTRACT

Many of the issues addressed with multi-agent approaches, such as distributed coordination and self-organization, are now becoming part of industrial and business systems. However, Multiagent Systems (MASs) are still not widely adopted in industry owing to the lack of a connection between MAS and software engineering. Since 2000, there is an effort to bridge this gap and to produce software engineering techniques for agent-based systems that guide the processes of design, development and maintenance. In Brazil, Agent-oriented Software Engineering (AOSE) was first investigated by the research group in the Software Engineering Laboratory (LES) at PUC-Rio, which after one decade of study in this area has built an AOSE community. This paper presents the history of AOSE at LES by discussing the sub-areas of MAS Software Engineering research and development that have been focus of the LES research group. We give examples of relevant results and present a subset of the extensive literature the group has produced during the last decade. We also report how we faced the challenges that emerged from our research by organizing and developing a research community at the intersection of software engineering, programming and MASs with a concern for scalability of solutions.

1. Introduction

Our motivation in writing the present paper is to put into perspective the contributions of our research group in the creation and development of the new area of Software Engineering of Multiagent Systems (MASs). We discuss our work starting in the year 2000 to allow comparison with other pioneering initiatives in the area. This viewpoint is consistent with research in this area as the first International Workshop on Agent-oriented Software Engineering (AOSE) took place in Limerick, Ireland, in 2000 and its post-proceedings were published by Springer in 2001 (Wooldridge and Ciancarini, 2001).

In this paper we use the acronym LES (Software Engineering Laboratory) to refer to our research group. LES is part of the Computer Science Department of PUC-Rio, Brazil. It was created in 1988 supported by a significant grant from IBM and focused its early efforts on the development of Object Oriented Systems. In the mid 90s, the research emphasis switched to the engineering of Web-based applications.

Since its inception in the late eighties, LESs focus has been to develop theoretical and experimental work on Software Engineering always oriented toward the solution of real-life large-scale and socially relevant applications. During the nineties our group at LES chose to focus on the education and e-commerce problem domains. Agents as powerful abstractions and MASs as an expressive modeling notation were applied to numerous case studies sometimes leading to spin-off companies of the laboratory.

Being familiar with the foundations of Software Engineering (SE) and Artificial Intelligence (AI), our group started amalgamating the two disciplines. Around 1998 the term MASs became a synonym for distributed AI (Weiss, 1999) and it started to become clear that the state-of-the-art of this new AOSE area was missing some of the key concepts existing in known practices for developing complex distributed systems.

Object-oriented software engineering has proven to be a powerful paradigm for supporting the development of high-quality software systems. However, we have been faced with the task of engineering large-scale systems composed of collaborating software components. These components achieve system goals through the coordination of autonomously distributed specialized services that exist in non-deterministic and open environments, like the Internet. This need for autonomous and pervasive behavior has spurred the re-examination of the notions and properties associated with software agents and MASs. MAS concepts cover most of the software development lifecycle from conceptual modeling and requirements specifications to architectural definition, design, implementation and deployment.

* Corresponding author.
E-mail addresses: lucena@inf.puc-rio.br (C. Lucena), ionunes@inf.puc-rio.br (I. Nunes).

0164-1212/$ – see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jss.2012.09.016

MAS and the underlying related theories provide natural support for properties such as autonomy, mobility, environment heterogeneity, organization, openness, and intelligence. As a consequence, agent-based systems have provided new insights into the complexity of developing and maintaining modern software. Although there are many methods and techniques for working with individual agents or systems built using only a few agents, developing robust large-scale agent-based systems calls for the creation of new software engineering approaches.

MAS features are now being applied to the development of large industrial and business systems, where such systems involve hundreds, or perhaps thousands of agents. Thus, there is a pressing need for software engineering methods to guide the process of MAS development and to allow the characteristics of these systems to be effectively managed. Without adequate development methods such systems will not be sufficiently dependable, robust, trustworthy and extensible. In addition agent-based systems will be difficult to comprehend and their components will not likely be reusable.

The complexity associated with large MAS is not straightforward. When a huge number of agents interact across heterogeneous environments, various phenomena occur that are not as easy to explain as when only a few agents collaborate. As multiple software agents cooperate in networked environments, they must be context-aware and deal with environmental uncertainty. Such interaction and the need for sensitivity to surroundings make coordination and management more difficult and increase the likelihood of the occurrence of exceptional situations in areas such as security, role assignment, privacy violation and other unexpected global effects. Moreover, as users and software engineers delegate more autonomy to their MAS, and put more trust in the results, new concerns can arise in real-life applications. Many existing agent-oriented solutions are far from ideal; in practice, the 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 MAS applications are to be commercially successful, MAS approaches will require scalable solutions based on software engineering approaches to ensure effective deployment and reuse.

Given this motivation for AOSE and also its many associated difficulties, there are plenty of reasons to pursue research in this area. In fact, the increasing cooperation and convergence of various kinds of computing entities, e.g. laptops, cellphones and personal digital assistants, is fundamentally changing the way we view computers and software (Garcia et al., 2003). The size, increasing complexity and potential for future applications to change the way society operates (e.g. community support, collaborative work and supervision) make centralized and direct control by a programmer nearly impossible (Choren et al., 2005). It is natural to delegate more autonomy and initiative to various software modules and to provide them with the ability to cooperate. MASs have been proposed as a conceptual framework to help design and construct such large scale autonomous and cooperative computing systems.

We provide in this paper three main contributions. First, as part of the celebrations of the 25th year of the Brazilian Software Engineering Symposium (SBES), we describe the emergence and evolution of this important research area in the context of software engineering — Agent-oriented Software Engineering. As, in Brazil, this area has its roots and is still mainly addressed by our laboratory, we focus on the research work developed by LES. Moreover, by presenting many works on this challenging and exciting research area, we aim at motivating (Brazilian) researchers to become part of this community, and make it stronger. And, finally, as we describe how this area was initiated in Brazil, we provide guidance for researchers with new ideas to promote new relevant research areas.

This paper is organized as follows. In Section 2, we provide a brief background on MAS and AOSE. In Section 3, we discuss the subareas of AOSE research and development that have been our group's focus since the year 2000. We give examples of relevant results and present a subset of the extensive literature we have produced during the last decade. Section 4 reports on how we managed to face the challenges presented in the last two sections by organizing and developing a research community at the intersection of software engineering, programming and MASs with a concern for scalability of solutions. Section 5 provides further relevant discussions about AOSE in LES. Finally we present our conclusions in Section 6.

2. Short background on MAS and AOSE

Multiagent Systems synthesize contributions from different areas, including artificial intelligence, software engineering and distributed computing to address the development of complex and distributed systems (Zambonelli et al., 2000). An agent is an abstraction that enjoys mainly the following properties (Wooldridge and Jennings, 1995).

Autonomy: agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state.

Social ability: agents interact with other agents (and possibly humans) via some kind of agent-communication language.

Reactivity: agents perceive their environment and respond in a timely fashion to changes that occur in it.

Pro-activeness: agents do not simply act in response to their environment, they are able to exhibit goal-directed behavior by taking the initiative.

Because of these characteristics, agents are a natural metaphor to understand complex and distributed systems (Jennings, 2001), thus being a powerful abstraction for their development. In order to support the construction of MASs, Agent-oriented Software Engineering (AOSE) (Wooldridge and Ciancarini, 2000) has investigated development approaches, resulting in the proposal of many methodologies and processes (Wooldridge et al., 2000; Cossentino, 2005; Bresciani et al., 2004) and modeling languages (Silva and Lucena, 2007). This lead MASs to be seen as a software engineering paradigm, which addresses the development of systems that contain many dynamically interacting components, each with their own thread of control while engaging in complex, coordinated protocols. The main idea of this paradigm (Wooldridge and Ciancarini, 2000) is to decompose complex and distributed systems into agents with the characteristics described above. A main difference between an agent and an object is that the former encapsulates not only data (its state) and how behaviors are realized, but also the *behavior selection process* and when such behaviors are necessary.

Due to the properties that agents have and the environment in which they are situated, many new problems emerge (Luck et al., 2005) and must be solved, which are often tackled with abstractions that are inspired by their analogy with human societies and human reasoning. For example, as an agent is autonomous and possibly self-interested, one cannot guarantee that it is going to execute what is expected. Therefore, *trust and reputation* (Ramchurn et al., 2004) issues must be dealt. In order to achieve common goals, agent interactions must be coordinated. As a consequence, organizational concepts (Dignum, 2009) are essential to support this coordination, such as assigning roles to particular agents, establishing and enforcing norms or observing their emergency. This is particularly interesting when avoiding possible chaos in large-scale open environments, such as the Internet. Last, but not least, the agent behavior selection process can be as simple as a set of rules, or they can be provided with cognitive abilities, which are associated with learning and reasoning techniques. For instance, the belief-desire-intention (BDI) architecture (Rao and Georgeff,