Science of Computer Programming 78 (2013) 747–761

Contents lists available at SciVerse ScienceDirect

Science of Computer Programming

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

Multi-agent oriented programming with JaCaMo

Olivier Boissier[a,*], Rafael H. Bordini[b], Jomi F. Hübner[c], Alessandro Ricci[d], Andrea Santi[d]

[a] *ISCOD/LSTI, Ecole Des Mines 158 Cours Fauriel, 42024 Saint-Etienne Cedex, France*
[b] *INF-UFRGS, Federal University of Rio Grande do Sul, CP 15064, 91501-970 Porto Alegre RS, Brazil*
[c] *DAS-UFSC, Federal University of Santa Catarina, CP 476, 88040-900 Florianópolis SC, Brazil*
[d] *DEIS, Alma Mater Studiorum–Università di Bologna, Via Venezia 52, 47521 Cesena (FC), Italy*

ARTICLE INFO

Article history:
Received 7 March 2011
Received in revised form 19 August 2011
Accepted 6 October 2011
Available online 29 October 2011

Keywords:
Multi-agent oriented programming
Autonomous agents
Shared environments
Agent organisations

ABSTRACT

This paper brings together agent oriented programming, organisation oriented programming and environment oriented programming, all of which are programming paradigms that emerged out of research in the area of multi-agent systems. In putting together a programming model and concrete platform called JaCaMo which integrates important results and technologies in all those research directions, we show in this paper, with the combined paradigm, that we prefer to call "multi-agent oriented programming", the full potential of multi-agent systems as a programming paradigm. JaCaMo builds upon three existing platforms: *Jason* for programming autonomous agents, $\mathcal{M}$OISE for programming agent organisations, and CArtAgO for programming shared environments. This paper also includes a simple example that illustrates the approach and discusses some real-world applications that have been or are being developed with JaCaMo.

1. Introduction

Current trends in computer science are facing up the challenges of building distributed and open software systems operating in dynamic and complex environments, interacting with and acting on the behalf of humans. In this context, multi-agent technologies can provide concepts and tools that give possible answers to the challenges of practical development of such systems by taking into consideration issues such as decentralisation and distribution of control, flexibility, adaptation, trust, security, and openness.

At least four separate communities within the multi-agent research community have been dealing with specific dimensions of such practical software development, namely the research communities interested in agent oriented programming languages [1–3], interaction languages and protocols [4], environment architectures, frameworks and infrastructures [5], and multi-agent organisation management systems [6–9]. The results produced by these communities have clearly demonstrated the importance of each dimension for the development of complex and distributed applications.

Nevertheless, perhaps a bit surprisingly, currently the engineering of such systems is still hampered by the usage of programming approaches and related development platforms that are mainly focused on just *one* specific dimension, at best connected in an ad hoc way with the other ones. This means that developers currently have ad hoc programming solutions for the development of complex multi-agent systems, typically based on low-level implementation mechanisms, without suitable high-level abstraction and tools. However the benefits of adopting a comprehensive approach – integrating different dimensions – were recognised in the context of AOSE (agent oriented software engineering) and MAS modelling communities quite a long time ago [10–12].

* Corresponding author.
E-mail addresses: Olivier.Boissier@emse.fr (O. Boissier), r.bordini@inf.ufrgs.br (R.H. Bordini), jomi@das.ufsc.br (J.F. Hübner), a.ricci@unibo.it (A. Ricci), a.santi@unibo.it (A. Santi).

0167-6423/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.scico.2011.10.004

In this paper, we aim at bringing such a perspective down to the programming level, investigating a concrete programming model and platform that allows a comprehensive integration of three multi-agent programming dimensions, namely the agent, environment, and organisation levels, preserving a strong separation of concerns but, at the same time, exploiting such dimensions in a synergistic way. To this end, we took as a starting point three existing platforms that have been developed for years within the context of the research in the separate communities mentioned above, namely *Jason* [13] for programming agents, CArtAgO [14] for programming environments, and $\mathcal{M}$OISE [15] for programming organisations.[1] Then, beyond a simple technological integration, we analyse how to integrate the related programming (meta-)models so as to come up with an approach that allows programmers to take advantage of such connections in order to simplify the development of complex systems. The result and main contribution of this paper is the JaCaMo conceptual framework and platform, which provides high-level first-class support for developing agents, environments, and organisations in synergy.

The remainder of the paper is structured as follows. In Section 2, we provide an account of the basic ideas on agent oriented programming and *multi-agent oriented programming*, including a discussion of related work. Then, in Sections 3 and 4, we describe the JaCaMo approach, first presenting the programming (meta-)model on which the framework is based – highlighting how the agent, environment, and organisation dimensions relate to each other – and then describing the platform architecture, along with the support for JaCaMo applications at runtime. To clarify the use of JaCaMo in practice and illustrate some of its features, we present in Section 5 some applications that were developed using our approach. Finally, in Section 6, we provide concluding remarks and some perspectives for future work.

2. Agent oriented and multi-agent programming: the state of the art and challenges

The idea of agent oriented programming was initiated by Shoham [1] as a new programming paradigm combining the use of mentalistic notions (such as belief) for programming (individual) autonomous agents and a societal view of computation. In the 90's, most of the work centred on a few languages that had seen significant theoretical work but had limited use in practice, and mostly centred on developing individual agents whether for a multi-agent system context or not. The work in this area in the 00's changed this picture substantially by producing many different agent programming languages based on varied underlying formalisms and inspired by various other programming paradigms. Furthermore, many such languages were developed into serious programming approaches, with working platforms and development tools. This led to some of these languages having now growing user bases and being used in many AI and multi-agent systems courses. Rather than referring to all those languages individually, we point the interested reader to [2,3,16] where many such languages have been presented in detail or comprehensive references given.

The important contribution of agent oriented programming as a new paradigm was to provide ways to help programmers to develop autonomous systems. For example, agent programming languages typically have high-level programming constructs which *facilitate* (compared to traditional programming languages) the development of systems that are continuously running and reacting to events that characterise changes in the dynamic environments where such autonomous systems usually operate. It is not only the case that agents need to take on new opportunities or revise planned courses of action because of changes in the environment; also agent programming facilitates programming agent behaviour that is not only reactive but also proactive in attempting to achieve long-term goals. The features of agent programming also make it easier, again compared to other paradigms, for programmers to ensure that the agents behave in a way that in the agent literature is referred to as "rational". For example, if a course of action is taken in order to achieve a particular goal (typically explicitly represented in the agent state), if the agent realises that the goal has not been achieved we would not expect the agent not to take further action to achieve that goal on behalf of its human designer unless there is sufficient evidence that the goal can no longer be achieved or is no longer needed.

While agent oriented programming in some two decades made impressive progress in supporting the programming of that kind of autonomous behaviour, it still was not able to achieve all that was expected of it in the context of multi-agent systems. The point is that multi-agent systems are normally used to develop very complex systems, where not only are many autonomous entities present, but also they need to interact in complex ways and need to have social structures and norms to regulate the overall social behaviour that is expected of them and, equally important, a shared environment can be an important and efficient source of coordination means for autonomous agents. Fortunately, while all the technicalities of language constructs that would facilitate the programming of (possibly intelligent) autonomous agents were being dealt with by some researchers in the autonomous agents and multi-agent systems research community, other researchers in that area were focusing precisely on the social/interaction and environment aspects of the development of multi-agent systems. Interestingly, as with agent oriented programming (AOP), the other researchers also coined what could appear to be other (separate) programming paradigms: organisation oriented programming (OOP) [17,18], interaction oriented programming (IOP) [19], and environment oriented programming (EOP) [20].

While they could indeed be independent programming paradigms, it has turned out – as we show in this paper – that the combination of all these dimensions of a multi-agent system into a single programming paradigm with a concrete

[1] Although we recognise that agent *interaction* (e.g., communication protocols) is also an important dimension of first-class abstractions, this is not yet incorporated into our platform, but we discuss how this dimension is currently handled and how we plan to address this in future work at the end of this paper.