WHITESTEIN
Technologies

AAMAS 2008 FOSE Panel

Opportunities to Support Wide-spread Adoption of Software Agents

Dr. Monique Calisti, Whitestein Technologies AG

Whitestein Technologies AG | Pestalozzistrasse 24 | CH-8032 Zürich
Tel +41 44-256-5000 | Fax +41 44-256-5001 | http://www.whitestein.com

Introduction

Whitestein Technologies is a leading innovator in the area of software agent technologies and autonomic computing. Our offering includes various agent-based products, solutions, and services for selected applications and industries, as well as a comprehensive middleware for the development and operation of autonomic, self-managing, and self-organizing systems and networks.

Several years of success with real-world deployments have shown that the advanced capabilities of software agents have matured enough to enable the design and implementation of a new generation of enterprise solutions, which are able to optimize their processes and to flexibly adapt in real-time to changing and unforeseen run-time conditions and requirements.

This, however, can only be realized by:

- Providing solid agent methodologies, platforms, tools and products for enterprise-grade development and deployment of agent technology.
- Facilitating the understanding of agent technology in close combination with specific application-driven business requirements.
- Taking into account dynamics of markets and complementary technologies.

We are happy to contribute to the FOSE-MAS session and present our perspective on its topics by answering the main questions the organizers put forward. We hope that our answers will be the ground for further discussion in Portugal.

Questionnaire

What are the main aspects that hamper progress in software engineering and MAS?

Lack of substantial efforts and focus in:

- Programming languages
- Libraries (meaning availability of deployable software)

- Inverted precedence between techniques and methodologies: software techniques should come first (meaning earlier research and maturity level) than methodologies.

Why is state-of-the-art in MAS research and engineering insufficiently reflected in state-of-the-practice in complex distributed systems?

Essentially for two main reasons:

The "state-of-the-practice" as such does not typically need any academic/scientific blessing: it just happens. Particularly, in the modern information exchange landscape (open-source movements, community-centric development, etc.) a "normal user" will not bother too much with the labels or underlying conceptual frameworks. She will just make the best use of what is readily available to suit her needs. Given the above-mentioned bias of the agent-MAS research community away from delivering concrete and usable software incarnations of their ideas, the agent concepts end up being severely under-represented.

Secondly, the diversity of people background and agendas within what has become the "agent community", which is a strength in other instances, ends up hampering the possibility to focus and synergize efforts towards a more restricted set of goals. This, however, would typically be a pre-requisite to the effective delivery of practical, directly applicable results, particularly software deliverables.

What is the future for agent-oriented methodologies?

From a research point of view agent-oriented methodologies can keep on refining conceptual and procedural aspects concerning agent-oriented software engineering. On the other hand, we see little hope from an adoption point of view as long as not enough attention is devoted to the delivery of concrete languages, libraries and tools that can be used to build applications, with no pre-requisite knowledge of agent oriented methodologies. Only after that people will start feeling the need of methodological support to bring order in what they are already doing.

What are the strong and weak points of state-of-the-art agent programming languages?

In our opinion, there simply isn't yet enough work about agent programming languages from a software engineering point of view. This means that most of the scientific work one can find when looking for keywords such as "agent-oriented programming" does not really deal with programming language design issues and even less with the specification, realization and assessment of actual prototype programming languages.

What makes software engineering of MAS different from mainstream software engineering?

Firstly, and luckily, the differences are increasingly being reduced. The fact that both the problem definition and the principled solution architectures of a mainstream system of today are much more similar to a MAS than it was 10 years ago is a sign that the agent community successfully foresaw the evolution of modern software engineering.

However, the full MAS conceptual framework seems to still have more than that in stock. Some of the ever-standing challenges of software engineering (reducing the gap between business users and system architects, enhancing the unforeseen reuse of software components, building homeostatic self-management into software systems) will be, in our opinion, better addressed by relying on suitable concrete applications of MAS ideas.

What are the key research challenges for software engineering and MAS?

Research challenges should not be mandated or defined too narrowly, of course. Here, we only suggest a few directions.

In MAS, one can have different kind of entities:

- Autonomous, reactive software components, typically the agents.
- Non-autonomous, reactive software components, e.g., services, artifacts.
- Passive, representational software components, e.g., objects, data.

Which category should be typed, and how? Should there be the same type system for all these three categories? How would the type system, e.g., for agents look like? Why?

What is it that we have to do to promote industrial adoption of AOSE? Can we do that, and how?

Producing more tangible deliverables (software applications, languages, libraries, tools that are concrete and usable incarnations of agent ideas) is a general pre-requisite. Contrary to popular belief, industry representatives are not hostile towards specific terms (such as "agent") when a concrete solution is provided. Of course, if the sales pitch is totally lacking concrete technological grasp, buzzwords are all that's left. But good ideas that solve relevant problems are always welcome, and names become less relevant.

Another key factor is the availability of technology transfer mechanisms, institutions and ultimately, people: the current situation can be improved. A series of University-level measures can be set up (or improved) to produce graduates that, without any specific research profile, simply have the MAS ideas and techniques as part of their professional bag. At the PhD level, doctors should be able to properly introduce theoretical MAS research concepts in their work environment if/when they apply. What

seems to be missing the most is a (probably PhD-grade) category of people who could play the role of technology evangelist and technical leader in an innovative industrial setting. More PhD positions should be geared towards this more practical kind of consulting stance, resulting in people who can understand the theory and the research state-of-the-art, but also comprehend the actual business and technical environment they will be operating in.

What actions are required to advance research in software engineering and MAS?

We foresee three main action streams:

- Include in the research agenda the aspects, which are more software-centric (design, implementation and testing).
- Try to establish more liaisons with research communities that focus on programming languages, libraries and infrastructures (particularly for distributed systems).
- Try to better promote the agent-driven research programme within institutions and bodies funding and organizing the R&D broad context like the EU Commission in the ICT space.