Multiagent System Product Lines: Challenges and Benefits

Joaquin Peña
University of Seville
Spain
joaquinp@us.es

Michael G. Hinchey
NASA Goddard Space Flight Center
USA
Michael.G.Hinchey@nasa.gov

Antonio Ruiz-Cortés
University of Seville
Spain
aruiz@us.es

1 Introduction

On the one hand, the field of Software Product Lines (SPL), as described elsewhere in this special issue, covers all the software development lifecycle necessary to develop a family of products where the derivation of concrete products is made systematically and rapidly [10]. On the other hand, Agent-Oriented Software Engineering (AOSE) is a new software engineering paradigm that arose to apply best practice in the development of complex Multi-Agent Systems (MAS) by focusing on the use of agents, and organizations (communities) of agents as the main abstractions [7].

Following a rather false start, agent technology has begun to come into its own. With the advent of biologically-inspired, pervasive, and autonomic computing, the advantages of, and necessity of, agent-based technologies and MASs has become obvious. Unfortunately, current AOSE methodologies are dedicated to developing single MASs. Clearly, many MASs will make use of significantly the same techniques, adaptations, and approaches. The field is thus ripe for exploiting the benefits of SPL: reduced costs, improved time-to-market, etc., and enhancing agent technology in such a way that it is more industrially applicable.

We believe that there is much that can be achieved by combining the two approaches: applying the SPL philosophy for building a MAS will afford all of the advantages of SPLs and make MAS development more practical. Thus, the contribution of this work is twofold: stress the feasibility and benefits of what we call *Multi-Agent Systems Product Lines* (MAS-PL); and show readers the main research challenges in the development of MAS-PLs.

2 Feasibility of MAS-PL and benefits

The software process proposed in AOSE presents many similarities with the process followed in SPL for the first activities of the *domain engineering*, which is in charge of providing the reusable core assets that are exploited during the derivation of products, done at the *application engineering* [10]. Following the nomenclature used in [10], the activities, usually performed iteratively and in parallel, of domain engineering that present correlation with AOSE are:

Domain Requirements Engineering. Both approaches use models based on similar concepts: features in the case of SPLs, and system-goals in the case of AOSE [3, 4]. Both represent requirements observable by the end user. Both approaches use hierarchical diagrams where features/goals are decomposed into finer grain ones. However, SPL emphasizes the analysis of the scope of the SPL, i.e. the products inside it, and the analysis of common and variable features across the SPL, which is not carried out by AOSE. In [5, 9], a first step toward adapting system-goals to MAS-PL and documenting variability is shown.

Domain Design. Both approaches develop architecture-independent models that attempt to analyze how features and its variability can be materialized. In AOSE, role models are used with this purpose [12], and some approaches in SPL also propose the same approach [6, 11]. However, agent-focused models show additional information that is not needed in SPL-role models, such as the goals of the agents, or whether they are used to abstract IA techniques, while not showing how these role models can be reused for different products.

Domain Realization. Both approaches focus on designing a detailed architecture. In the case of SPL, a common architecture for all products and a set of reusable assets. In the case of AOSE, a single architecture that fulfills all of the system-goals of the MAS. Some approaches in both fields base the construction of the architecture on role model composition [6, 11]. In [9], authors presented the first steps toward building the core architecture of a MAS-PL based on automatic analysis of system-goals models adapted to feature models using [2].

This, along with the shown first research papers developed under this field, shows that the benefits of enabling MAS-PL are reachable. The main benefit is straight forward: AOSE can benefits of all SPL advantages helping it to reach the industrial world. However, as we show in the next section, there exist also a number of research challenges that must constitute the research agenda needed to allow MAS-PL to become a reality.

3 Future Challenges

SPL for distributed systems. Distributed systems have not been a hot topic in the SPL field. However, MASs are distributed systems that will need new adapted techniques to be covered. Although certainly it will affect to the whole development cycle, one of the first steps we foresee is the need for investing in the use of interaction-based models, such as role models. The research undertaken over this topic may extend the applicability of SPL not only to MASs, but also to other kind of distributed systems such as web services[1].

AOSE deficiencies. As shown before, AOSE does not cover some of the activities of SPL. These are mainly concentrated on *commonality analysis*, and its implications at whole SPL approach. Another hot topic can be found in the *product management* activity that is performed in parallel with domain and application engineering. It is in charge of managing the economic aspects of a SPL. Given that the products and the markets for MASs are quite different from the ones typically used in SPL, a deal of effort must be put on studying these aspects. Finally, as AOSE is devoted to develop single products, application engineering is not present in AOSE. Researchers should also invest efforts on studying this activity.

Management of evolving systems. Agent-based evolving systems results on large software systems that adapt and learns form changes in the environment. The development of these systems results in a complex task where systems usually become unmanageable from an engineering point of view. SPL can help this task by viewing an evolving system as a SPL where a different state in the system is viewed as a separate product [8]. This decomposes the system into well identified chunks and a well defined context where each product will appear, what helps to deal with the inherent complexity of MASs.

Self-* properties of agents We believe that agent technology may bring also advantages to SPL. Given research efforts invested in providing agents with the capabilities to communicate with each other at the semantic level, or to provide capabilities of self-organization, self-optimization, self-healing, etc., the maintenance and evolution of the core architecture may be simplified. Additionally, the integration costs of new features for a certain product, or even the entire SPL, may be decreased.

4 Conclusions

MAS-PL, drawing benefits from both SPL and AOSE, will help in the industrial exploitation of agent technology, saving both effort and cost. We have identified several challenges, such as adapting current AOSE engineering techniques to the SPL philosophy, which in many cases requires the development of new activities/models from scratch. However, a symbiosis between both AOSE and SPL arises when AOSE also provides benefits to SPL, mainly through encouraging and improving research on SPL of complex distributed systems. As can be seen, MAS-PLs represent a great, and worthwhile, challenge that will certainly attract the interest of many practitioners and researchers.

References

[1] D. Benavides, A. R. Cortés, M. A. Serrano, and C. M. de Oca. A first approach to build product lines of multi-organizational web based systems (mows). In T. Böhme, V. Larios-Rosillo, H. Unger, and H. Unger, editors, *IICS*, volume 3473 of *Lecture Notes in Computer Science*, pages 91–98. Springer, 2004.

[2] D. Benavides, A. Ruiz-Cortés, and P. Trinidad. Automated reasoning on feature models. *LNCS, Advanced Information Systems Engineering: 17th International Conference, CAiSE 2005*, 3520:491–503, 2005.

[3] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. Tropos: an agent-oriented software development methodology. *Journal of Autonomous agents and Multiagent Systems*, 8(3), 2004.

[4] K. Czarnecki and U. Eisenecker. *Generative Programming: Methods, Tools, and Applications*. Addison–Wesley, 2000.

[5] J. Dehlinger and R. Lutz. A product-line approach to promote asset reuse in multi-agent systems. In *SELMAS*, volume 3914 of *Lecture Notes in Computer Science*, pages 161–178. Springer, 2005.

[6] A. Jansen, R. Smedinga, J. Gurp, and J. Bosch. First class feature abstractions for product derivation. *IEE Proceedings - Software*, 151(4):187–198, 2004.

[7] N. Jennings. An agent-based approach for building complex software systems. *Communications of the ACM*, 44(4):35–41, 2001.

[8] J. Peña, M. G. Hinchey, and A. Ruiz-Cortés. Managing the evolution of an enterprise architecture using a mas-product-line approach. In *International Workshop on Worshop on System/Software Architectures 2006*, Nevada,USA, June, 2006. CSREA Press.

[9] J. Peña, M. G. Hinchey, and A. Ruiz-Cortés. Building the core architecture of a multiagent system product line: With an example from a future nasa mission. In *7th International Workshop on Agent Oriented Software Engineering 2006*, page to be published, Hakodate, Japan, May, 2006. LNCS.

[10] K. Pohl, G. Böckle, and F. van der Linden. *Software Product Line Engineering : Foundations, Principles and Techniques*. Springer, September 2005.

[11] Y. Smaragdakis and D. Batory. Mixin layers: an object–oriented implementation technique for refinements and collaboration-based designs. *ACM Trans. Softw. Eng. Methodol.*, 11(2):215–255, 2002.

[12] F. Zambonelli, N. R. Jennings, and M. Wooldridge. Developing multiagent systems: The gaia methodology. *ACM Trans. Softw. Eng. Methodol.*, 12(3):317–370, 2003.