By Joaquin Peña, Michael G. Hinchey, and Antonio Ruiz-Cortés

MULTI-AGENT SYSTEM PRODUCT LINES: CHALLENGES AND BENEFITS

On the one hand, the field of software product lines (SPLs), as described in the articles in this section, covers all the software development life cycle 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 the best practices in the development of complex multi-agent systems (MASs) by focusing on the use of agents, and organizations (communities) of agents as the main abstractions [7].

Following a somewhat slow 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 SPLs: reducing costs, improving time to market, and enhancing agent technology in such a way that it is more industrially applicable.

We believe there is much that can be achieved by combining the two approaches: applying the SPL philosophy for building a MAS will afford all the advantages of SPLs and make MAS development more practical. Thus, our intent in this article is twofold: to stress the feasibility and benefits of what we call multi-agent systems product lines (MAS-PL) and demonstrate the main research challenges in the development of MAS-PLs.

Feasibility of MAS-PL and Benefits

The software process proposed in AOSE presents many similarities with the process followed in SPLs 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 during 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 diagrams. However, the SPL emphasizes the analysis of the scope of the SPL, that is, 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 their variability can be materialized. In AOSE, role models are used for this purpose [12], and some approaches in SPL also propose the same approach [6, 11]. However, agent-focused models show additional information not needed in SPL role models, such as the goals of the agents, or whether they are used to abstract information architecture 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 the 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 first research papers developed in this field, shows that the benefits of enabling MAS-PL are obtainable. The main benefit is straightforward: AOSE can help SPL gain increased acceptance in the industrial world. However, as we show here, a number of research challenges exist that must constitute the research agenda required for MAS-PL to become a reality.

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 for this topic may extend the applicability of SPL not only to MASs, but also to other kinds of distributed systems such as Web services [1].

AOSE Deficiencies. As shown previously, AOSE does not cover some of the activities of SPL. These are mainly concentrated on commonality analysis, and its implications for the SPL approach. Another important topic involves 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 the products and the markets for MASs are quite different from the ones typically used in SPLs, a great amount of effort must be made toward 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 result in large software systems that adapt and learn from 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, which helps to deal with the inherent complexity of MASs.

Self-* properties of agents. We believe agent technology may also bring 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, and self-healing, 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.

Conclusion

MAS-PL, incorporating 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 and 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. C

References

1. Benavides, D., Ruiz-Cortés, A., Serrano, M.A., and de Oca, C.M. A first approach to build product lines of multi-organizational Web-based systems (MOWS). In T. Böhme, V. Larios-Rosillo, and H. Unger, Eds., IICS, volume 3473 of *Lecture Notes in Computer Science*, Springer, 2004, 91–98.
2. Benavides, D., Ruiz-Cortés, A., and Trinidad, P. Automated reasoning on feature models. In *Proceedings of the Advanced Information Systems Engineering 17th International Conference* (*CaiSE 2005)*, LNCS 3520, 2005, 491–503.
3. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., and Mylopoulos, J. Tropos: An agent-oriented software development methodology. *Journal of Autonomous Agents and Multiagent Systems 8*, 3 (2004).
4. Czarnecki, K. and Eisenecker, U. *Generative Programming: Methods, Tools, and Applications*. Addison–Wesley, 2000.
5. Dehlinger, J. and Lutz, R. A product-line approach to promote asset reuse in multi-agent systems. In SELMAS, volume 3914 of *Lecture Notes in Computer Science*, Springer, 2005, 161–178.
6. Jansen, A., Smedinga, R., Gurp, J., and Bosch, J. First class feature abstractions for product derivation. *IEEE Proceedings—Software 151*, 4 (2004), 187–198.
7. Jennings, N. An agent-based approach for building complex software systems. *Commun. ACM 44*, 4 (2001), 35–41.
8. Peña, J., Hinchey, M.G., and Ruiz-Cortés, A. Managing the evolution of an enterprise architecture using a MAS-product line approach. In *Proceedings of the International Workshop on System/Software Architectures 2006*. CSREA Press, 2006.
9. Peña, J., Hinchey, M.G., and Ruiz-Cortés, A. Building the core architecture of a multiagent system product line: With an example from a future NASA mission. In *Proceedings of the ACM 7th International Workshop on Agent Oriented Software Engineering* (Hakodate, Japan, May 2006), 13–24.
10. Pohl, K., Böckle, G., and van der Linden, F. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer, 2005.
11. Smaragdakis, Y. and Batory, D. Mixin layers: An object–oriented implementation technique for refinements and collaboration-based designs. *ACM Transactions on Software Engineering Methodology 11*, 2 (2002), 215–255.
12. Zambonelli, F., Jennings, N.R., and Wooldridge, M. Developing multiagent systems: The GAIA methodology. *ACM Transactions on Software Engineering Methodology 12,* 3 (2003), 317–370.

Joaquin Peña (joaquinp@us.es) is a lecturer in the Department of Computer Science Languages and Systems at the University of Seville in Spain.

Michael G. Hinchey (Michael.G.Hinchey@nasa.gov) is the director of the Software Engineering Laboratory at the NASA Goddard Space Flight Center in Greenbelt, MD.

Antonio Ruiz-Cortés (aruiz@us.es) is an associate professor in the Department of Computer Science Languages and Systems at the University of Seville in Spain.

The work reported in this article was supported by the Spanish Ministry of Science and Technology under grants TIC2003-02737-C02-01 and TIN2006-00472 and by the NASA Software Engineering Laboratory, NASA Goddard Space Flight Center, Greenbelt, MD, USA.