The Software Engineering of Agent-Based Intelligent Adaptive Systems

Leon Sterling Thomas Juan

Department of Computer Science & Software Engineering,

University of Melbourne, Victoria, 3010, Australia

{leon, tlj}@cs.mu.oz.au

ABSTRACT

Future software systems will be intelligent and adaptive. They will have the ability to seamlessly integrate with smart applications that have not been explicitly designed to work together. Traditional software engineering approaches offer limited support for the development of intelligent systems. To handle the tremendous complexity and the new engineering challenges presented by intelligence, adaptiveness and seamless integration, developers need higher-level development constructs. Agent concepts are natural to describe intelligent adaptive systems. Agent-based technologies have been incorporating software engineering practices, and have matured to offer useful insights and concrete practices to mainstream software engineers.

This tutorial presents the state of the art in agent development from a software engineering perspective, focusing on practices that are applicable today. We will walk the audience through analysis, design and verification of a portion of a real-world problem: a Smart Home Network. We show how agent concepts more naturally match the engineering challenges of such systems like trust between adaptive components. The audience will have hands-on experience with analyzing and designing parts of the Smart Home Network and learn how to incorporate agent technologies into their current projects.

Categories and Subject Descriptors

D.2.10 [**Software Engineering**]: Design – *methodologies.*
I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence – *intelligent agents, multiagent systems.*

General Terms: Design, Languages.

Keywords: ROADMAP, Intelligent adaptive systems, software engineering, intelligent agent and multiagent systems.

1. INTELLIGENT ADAPTIVE SYSTEMS

Intelligent systems are able to act rationally to seek optimal solutions for their design objectives. From the user perspective, this means delivering context-aware human-like services with minimal user effort. Often, what seems rational to one user is not rational to another. Therefore, the system may need to reason about and adapt to individual users and modify its behaviour accordingly. As a natural extension of intelligence, such systems should seamlessly integrate with other applications that have not been explicitly designed to work together. In the example of a Smart Home Network, the system should provide customized services that are rational to the actual family it's serving, and learn to accommodate their particular needs.

Such systems represent a significant increase of program complexity and raise new engineering challenges. Since what is *rational* to a *particular* user cannot be determined before deployment, the correct behaviour of the system cannot be frozen during analysis and hard-wired during design and implementation. Instead, mechanisms must be built into the applications to dynamically discover what constitutes rational services to each user and dynamically generate the services. To enable the applications to adapt in a way that is traceable and maintains system quality, large amounts of explicit knowledge and explicit reasoning are needed for generating the rational services. Here, conventional software engineering techniques fail to support the engineering of explicit knowledge and reasoning. Software quality goals become fuzzy too, for example, different users have different interpretation of usability or security on different occasions. Other issues such as "how to handle conflicts between adaptive components that weren't designed and tested together?" or "how to assure quality goals are not compromised as the system changes its behaviours dynamically?" are also not addressed by conventional software engineering techniques. To effectively address these high-level issues, high-level development constructs are needed.

2. THE ROADMAP APPROACH

The area of Agent Oriented Software Engineering (AOSE) has benefited from recent advances of software engineering research and matured to offer useful insight and concrete practices back to mainstream software practitioners. In addition to presenting a broad survey on various agent-based techniques and technologies, this tutorial focuses on the ROADMAP approach [1,2,3,4] to illustrate and address the issues and engineering challenges in developing intelligent adaptive systems in more depth.

Agent concepts in ROADMAP are shown in Fig. 1. Roles and protocols are similar to classes and method definitions in OO, representing the software interface, while agents and services are similar to objects and method bodies, representing concrete implementation. A key difference in our approach is that class and method definitions are abstract in OO and can not be changed without re-compilation, while Roles and Protocols are concrete entities at runtime, allowing the system interfaces (or the system architecture) to be reasoned about and modified dynamically.

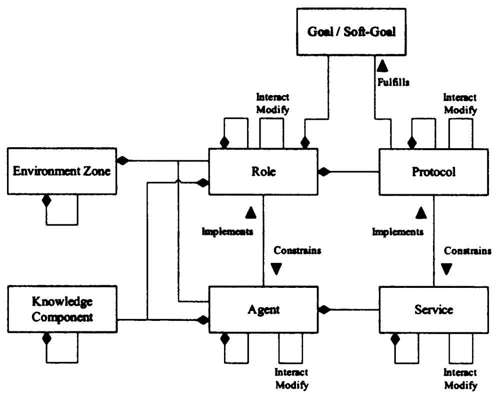

Figure 1. ROADMAP agent concepts

Agents interact by message passing through their roles (interface for agents), and messages between roles are organized into protocols, which captures the control and data flow in the system (notation based on UML 2). Having explicit protocols avoids the OO practice of hiding / hard-wiring data and control flow in method calls made in the bodies of methods and makes understanding or debugging the system easier. Since agents do not interact directly, by changing the roles and protocols, control and data flow in the system could be easily re-directed at runtime without affecting the underlying implementation (agent/service). This notion of *dynamic interfaces* makes software architecture flexible at runtime. Third party qualifiers such as [Encryption, RSA, 1024-bits] can be applied to protocols. Execution infrastructures can load the qualifiers and run the appropriate methods before, after or during the passing of a message (similar to and more flexible than aspect oriented programming). By explicitly binding agents and services with abstract knowledge and goals at runtime, we allow users to configure or modify program structure and behaviour dynamically through high-level knowledge languages. In future, naive users will be able to safely customize their applications without technical knowledge in a fashion similar to building with Lego blocks.

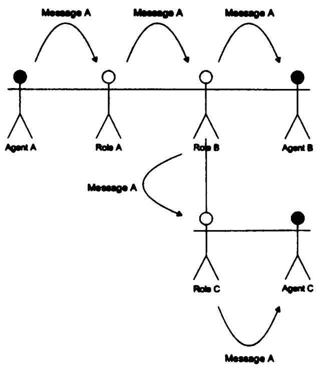

Figure 2. Message sending via roles

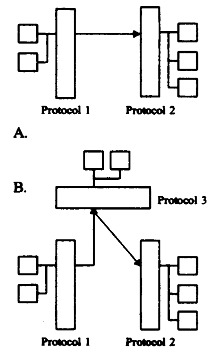

Figure 3. Re-routing interaction in the system

The ROADMAP methodology provides practical guidance for industry developers to utilize the agent concepts and technologies. The high-level agent constructs are natural for modeling intelligent behaviours such as trust and learning. The developer is able to model intelligent adaptive systems as "human-like" agent organizations, and consistently refine the high-level models to lower level models so various execution technologies can be used for implementation. As an example, ROADMAP extends work on Goal Oriented Requirements Engineering to incorporate goals in analysis, design, implementation and verification and validation. (See a Smart Home Intruder Handling Scenario in Figure 4 and 5)

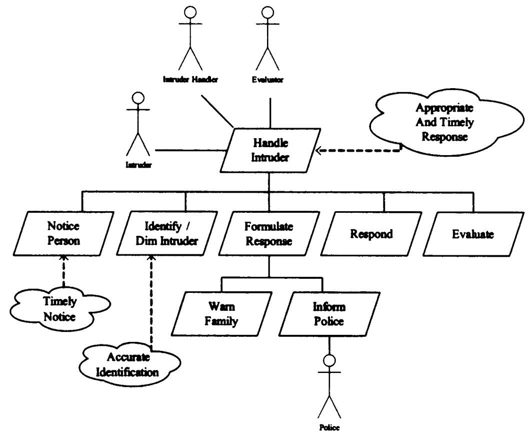

Figure 4. Goals and roles in ROADMAP analysis

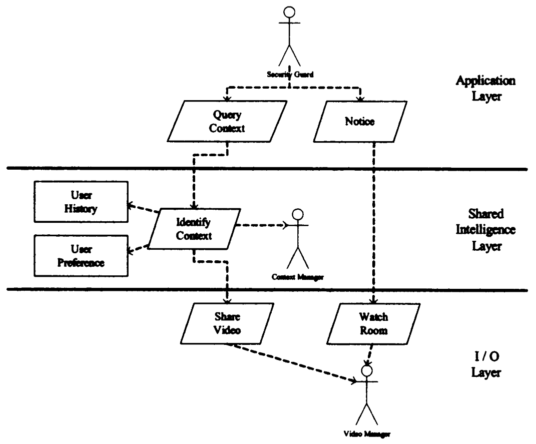

Figure 5. Goals, roles and knowledge components interact in a layered architecture in ROADMAP design

3. REFERENCES

[1] Juan, T. and Sterling, L., "Achieving Dynamic Interfaces with Agent Concepts", Proc. of the 3rd Int. Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2004), p688-695, New York, USA, July 2004.

[2] Juan, T., Sterling, L., Martelli, M. and Mascardi, V., Customizing AOSE Methodologies by Reusing AOSE Features , Proc. of the 2nd Int. Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2003), p113-120, Melbourne, Australia, July 2003.

[3] Juan, T. and Sterling, L., "The ROADMAP Meta-model for Intelligent Adaptive Multi-Agent Systems in Open Environments", Proc. of the 4th International Workshop on Agent Oriented Software Engineering, at AAMAS'03, p53-68, Springer-Verlag, Melbourne, Australia, July 2003.

[4] Juan, T., Pearce, A. and Sterling, L., ROADMAP: Extending the Gaia methodology for Complex Open Systems, Proc. of the 1st Int. Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2002), p3-10, Bologna, Italy, July 2002