On the Use of Patterns in Agent System Design

Michael Weiss[1]

[1] School of Computer Science, Carleton University, Ottawa, Canada
weiss@scs.carleton.ca

1 Introduction

In this paper we make a case for a pattern-driven approach to agent system design that complements the goal-driven approach of most design methodologies.

Current approaches to agent system design such as Gaia [15], and MASSIVE [11], are generally goal-driven. This is understandable given the nature of agents as goal-driven entities. Therefore, in these design approaches, an agent system is designed by iteratively decomposing system goals until they can be assigned to individual agents.

However, this may lead developers to solve the same design problems over and over without benefiting from how they were resolved in the past, resulting in duplicated effort, and inconsistent designs. A more effective and less ad hoc approach is to build an agent system incrementally from well-documented interaction patterns.

In the sections below we introduce the notions of patterns and agent patterns, and discuss the role of patterns in agent system design. Following that we outline the steps of a pattern-based methodology, and relate it to our current work on formalizing agent patterns using the non-functional requirements (NFR) framework [5].

2 Agent Patterns

Patterns are reusable solutions to recurring design problems, and provide a vocabulary for communicating these solutions to others. The documentation of a pattern goes beyond documenting a problem and its solution. It also describes the *forces* or design constraints that give rise to the proposed solution [1]. These are the undocumented and generally misunderstood features of a design. Forces can be thought of as pushing or pulling the problem towards different solutions. A good pattern balances the forces.

Patterns are not used in isolation. Although individual patterns are useful at solving specific design problems, we can benefit further from positioning them among one another to form a *pattern language*. Each pattern occupies a position in a network of related patterns, in which each pattern contributes to the completion of patterns "preceding" it in the network, and is completed by patterns "succeeding" it.

A pattern language guides developers through the process of generating a system. Beck and Johnson [4] describe this generative quality of patterns: "Describing an architecture with patterns is like the process of cell division and specialization that drives growth in biological organisms. The design starts as a fuzzy cloud representing the

system to be realized. As patterns are applied to the cloud, parts of it come into focus. When no more patterns are applicable, the design is finished."

There is by now a growing literature on the use of patterns to capture common design practices for agent systems [3, 8, 7]. Aridor and Lange [3] describe a set of domain-independent patterns for the design of mobile agent systems. They classify mobile agent patterns into traveling, task, and interaction patterns. Kendall et al [8] capture common building blocks for the internal architecture of agents in patterns.

Deugo and Weiss [7] identify a set of patterns for agent coordination, which are, again, domain-independent. They classify agent patterns into architectural, communication, traveling, and coordination patterns. They also describe an initial set of global forces that push and pull solutions for coordination. Kendall [9] reports on work on a domain-specific catalog of patterns developed at BT. Weiss [14] describes a pattern language for agent-based e-commerce. In related work, Kolp and Giorgini [10] document organizational styles for multi-agent systems using the Tropos framework.

The separate notion of an *agent pattern* can be justified by differences between the way agents and objects communicate, their level of autonomy, and social ability [6]. Agent patterns are documented in a similar manner as patterns, except for the structure of an agent pattern where we will make use of role models [13, 9] instead of collaboration diagrams. The distinction between role models and collaboration diagrams is the level of abstraction: a collaboration diagram shows the interaction of instances, whereas a role model shows the interaction of roles to be filled.

3 Pattern-driven Agent System Design

Instead of proceeding from high-level goals and arriving at an implementation through iterative refinement, in a pattern-driven approach we start from proven solutions, and *compose* our system by systematically instantiating patterns. The goal-driven and pattern-driven approaches to design are, of course, complementary. Our experience building agent systems suggests that the design approach that proceeds in both a top-down and a bottom-up direction in parallel will lead to the best results

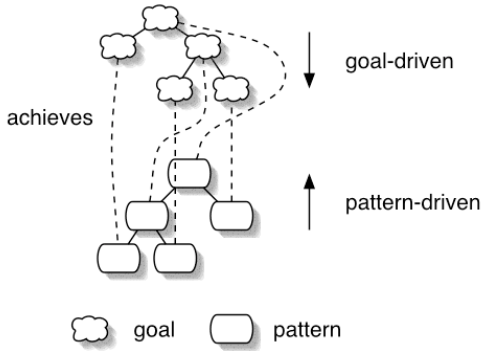

Fig. 1. Complementarity of the pattern-driven and goal-driven approaches

The implementation of the pattern-directed approach involves the following steps. We are currently building a tool that assists the designer with this process.

Identify Domain Forces. For a given domain identify the core design trade-offs (called forces in patterns) that push and pull the design into different directions. For example, for the e-commerce domain, these include information overload, search costs, privacy, ensuring quality, and identity (see [14] for more detail). In addition, there are forces motivating the use of agents (such as autonomy, need to interact, multiple interfaces, and adaptability) to be considered for all domains.

Document Roles. Document the roles and their subtypes used in the pattern language. Individual patterns document how these roles interact in a given design context. The task of the designer is, by selecting patterns, to assign roles to agents. In the example of the e-commerce domain, we identified four top-level roles as User, Task, Service, and Resource; these should be applicable to other domains, as well.

Document Patterns and their Dependencies. Document the patterns and their dependencies in the form of a pattern language. Each pattern should document the forces it helps resolve, and how instantiating the pattern will change the system. This includes the resulting role model and the forces that still need to be resolved. Semi-formal methods can be used to document the forces in a pattern. In Araujo and Weiss [2] we have investigated the use of the NFR framework to document patterns.

Identify the Overall Design Goals. Identify the overall design goals, both functional and non-functional. Generally, the identification of patterns based on merely functional goals is rather straightforward. However, although multiple patterns may satisfy the *same* functional goals, their implications on the design in terms of non-functional goals must be carefully considered. The main thrust of our semi-formal pattern representation is geared towards matching on non-functional goals.

Select Patterns. In a first pass, select patterns based on how well they match the functional goals of the system, and then refine the selection by considering non-functional design goals. Compare the patterns and rank them on basis of their compatibility with these goals. In our tool we will use the algorithm described in McPhail and Deugo [12]. Repeat this step until all forces have been resolved.

4 Conclusion

In this paper we described a pattern-driven approach to agent system design. This approach is complementary to the goal-driven approach that most published methodologies use. The combined design approach proceeds, in parallel, in a top-down (goal-driven) and a bottom-up (pattern-driven) direction.

Work on agent patterns is still at an initial stage. Before patterns can play the same role for agent system design as in the object-oriented world, further work is required. One problem is that only a relatively small number of agent patterns have been documented to date, when compared to the number of object patterns.

More work is required to document the forces that govern agent system design, and to understand their interactions. A related open problem is the selection of a pattern that is compatible with the stated functional and non-functional goals of an application. We are working on a new pattern representation using the NFR framework.

References

1. Alexander, C., A Pattern Language, Oxford University Press, 1977
2. Araujo, I., and Weiss, M., Using the NFR Framework for Representing Patterns, submitted to Pattern Languages of Programming (PLoP-02), 2002
3. Aridor, Y., Lange, D., Agent Design Patterns: Elements of Agent Application Design, Second Intl. Conference on Autonomous Agents, IEEE, 1998
4. Beck, K., and Johnson, R., Patterns Generate Architectures, European Conference on Object Oriented Programming (ECOOP-94), 139-149, 1994
5. Chung L., Representing and Using Non-Functional Requirements: A Process-Oriented Approach, Department of Computer Science University of Toronto, 1993
6. Deugo, D., Oppacher, F., et al., Patterns as a Means for Intelligent Software Engineering, Intl. Conference on Artificial Intelligence (IC-AI 99), CSREA Press, 605-611,1999
7. Deugo, D., Weiss, M., and Kendall, L., Reusable Patterns for Agent Coordination, in: Omicini, A., et al (eds.), Coordination of Internet Agents, Springer, 2001
8. Kendall, E., Murali Krishna, P., Pathak, C. et al, Patterns of Intelligent and Mobile Agents, Second Intl. Conference on Autonomous Agents, IEEE, 1998
9. Kendall, E., Role Models: Patterns of Agent System Analysis and Design, Agent Systems and Applications/Mobile Agents (ASA/MA-99), ACM, 1999
10. Kolp, M., Giorgini, P., Mylopoulos, J., A Goal-Based Organizational Perspective on Multi-Agent Architectures, Eighth Intl. Workshop on Agent Theories, Architectures, and Languages (ATAL-2001), 2001
11. Lind, J., Iterative Software Engineering for Multi-Agent Systems: The MASSIVE Method, LNCS 1994, Springer, 2001
12. McPhail, J.C., and Deugo, D., Deciding on a Pattern, 14th Intl. Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE-01) LNCS 2070, Springer, 2001
13. Riehle, D., and Gross, T., Role Model Based Framework Design and Integration, Conference on Object- Oriented Programs, Systems, Languages, and Applications (OOPSLA-98), 117-133, ACM, 1998
14. Weiss, M., Patterns for e-Commerce Agent Architectures: Using Agents as Delegates, Pattern Languages of Programming (PLoP-01), 2001
15. Wooldridge, M., Jennings, N., and Kinny, D., The Gaia Methodology for Agent-oriented Analysis and Design, Journal of Autonomous Agents and Multi-Agent Systems, 2002