Chapter XI
Patterns for Designing Agent-Based E-Business Systems

Michael Weiss
Carleton University, Canada

ABSTRACT

Agents are rapidly emerging as a new paradigm for developing software applications. They are being used in an increasing variety of applications, ranging from relatively small systems such as assistants to large, open, mission-critical systems like electronic marketplaces. One of the most promising areas of applications for agent technology is e-business. In this chapter, we describe a group of architectural patterns for agent-based e-business systems. These patterns relate to front-end e-business activities that involve interaction with the user, and delegation of user tasks to agents. Patterns capture well-proven, common solutions, and guide developers through the process of designing systems. This chapter should be of interest to designers of e-business systems using agent technology. The description of the patterns is followed by the case study of an online auction system to which the patterns have been applied.

INTRODUCTION

Agents are rapidly emerging as a new paradigm for developing software applications. They are being used in an increasing variety of applications, ranging from relatively small systems such as assistants to large, open, mission-critical systems like electronic marketplaces. One of the most promising areas of applications for agent technology is e-business (Papazoglou, 2001). In this chapter, we describe a group of architectural patterns for agent-based e-business systems. These patterns relate to front-end e-business activities that involve interaction with the user, and delegation of user tasks to agents.

The chapter is structured as follows. First, we provide a background on patterns and their application to the design of agent systems. Then, we discuss the forces or design constraints that need to be considered during the design of agents for e-business systems. This is followed by a description of the agent patterns for e-business. A number of examples illustrate the application of these patterns. Finally, we discuss current trends and opportunities for future research and offer concluding remarks.

BACKGROUND

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 (Alexander, 1979). 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 these forces. A set of patterns, where one pattern leads to other patterns that refine or are used by it, is known as a pattern language. A pattern language can be likened to a process: it guides designers who wants to use those patterns through their application in an organic manner. As each pattern of the pattern language is applied, some of the forces affecting the design will be resolved, while new unresolved forces will arise as a consequence. The process of using a pattern language in a design is complete when all forces have been resolved.

There is by now a growing literature on using patterns to capture common design practices for agent systems. Aridor and Lange (1998) describe domain-independent patterns for the design of mobile agent systems. They classify mobile agent patterns into traveling, task, and interaction patterns. Kendall, Murali Krishna, Pathak, et al. (1998) use patterns to capture common building blocks for the architecture of agents. They integrate these patterns into the layered agent pattern, which serves as a starting point for a pattern language for agent systems based on the strong notion of agency. Schelfthout, Coninx, et al. (2002), on the other hand, document agent implementation patterns suitable for developing weak agents.

Deugo, Weiss, and Kendall (2001) 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. Kolp, Giorgini, and Mylopoulos (2001) document domain-independent organizational styles for multi-agent systems using the Tropos methodology. Weiss (2004) motivates the use of agents through a set of patterns that document the forces involved in agent-based design and key agent concepts.

On the other hand, Kendall (1999) reports on work on a domain-specific pattern catalog developed at BT Exact. Several of these patterns are documented using role models in a description of the ZEUS agent building kit (Collis & Ndumu, 1999). Shu and Norrie (1999) and the author in a precursor to this chapter have also documented domain-specific patterns, respectively, for agent-based manufacturing and electronic commerce. However, unlike most other authors, they present the patterns in the form of a pattern language. This means that the relationships between the patterns are made explicit in such a way that they guide a developer through the process of designing a system.

Lind (2002) and Mouratidis, Weiss, and Giorgini (2006) suggest that we can benefit from integrating patterns with a development process, while Tahara, Oshuga, and Hiniden (1999) and Weiss (2003) propose pattern-driven development processes. Lind (2002) suggests a view-based categorization scheme for patterns based on the MASSIVE methodology. Mouratidis et al. (2006) document a pattern language for secure agent systems that uses the modeling concepts of the Tropos methodology. Tahara et al. (1999) propose a development method based on agent patterns and distinguish between macro and micro architecture patterns. Weiss (2003) documents a process for mining and applying agent patterns.