An Agile Toolkit to Support Agent-Oriented and Service-Oriented Computing Mechanisms

Asif Qumer and Brian Henderson-Sellers

Faculty of Information Technology, University of Technology, Sydney
2000 Broadway, NSW, Australia
{asif,brian}@it.uts.edu.au

Abstract. The complex nature of the software development paradigm and the rapid acceptance of emerging abstraction mechanisms, such as agent-oriented and service-oriented computing, highlight the increasing need for re-evaluation of existing software development approaches that focus on agile software development methodologies (primarily originating in object-oriented technology); since existing object-oriented, structure-oriented and component-oriented approaches embodied in an agile approach cannot be applied immediately to agent and service-oriented computing. Therefore, we present here, an agile toolkit (Java-based) to facilitate the construction of multi-abstraction or m-abstraction situation-specific agile processes for software development projects. This paper only presents the newly emergent abstraction concepts of agent and service, and does not discuss the well-established object-oriented mechanism used in current agile approaches.

Keywords: Agile methods, Agent-oriented, Service-Oriented, Method Engineering, M-abstraction.

1 Introduction

Agile software development methods mainly focus on object-oriented software technology [20] and lack the support for other emerging abstraction mechanisms such as agent-oriented and service-oriented computing. According to Luck *et al.* [23], the object-oriented paradigm is not immediately suitable for the development of multiagent systems because it does not address the autonomous behaviour of agents. The concepts of agents and objects are not the same. An agent is an autonomous, interactive, communication-focused and flexible complex entity (existing in some environment) that cooperates with other agents to solve a complex problem [18], [20], [36]. In addition, the concept of service in the agent-oriented context is dissimilar to the concept of services of an object. The emerging concept of service-oriented computing demands the re-evaluation of software development methodologies because the existing object-oriented, component-oriented or structure-oriented analysis and design methods are not immediately applicable for the development of service-oriented applications [13]. Service-oriented computing is a further higher-level abstraction transition from objects and distributed components [12].

J. Münch and P. Abrahamsson (Eds.): PROFES 2007, LNCS 4589, pp. 222–236, 2007.

There are several agile and agent-oriented methods whilst very few service-oriented application development methods have been reported; and none, to the best of our knowledge, that integrate multiple paradigmatic architectures. Indeed, agent-oriented and service-oriented methods are not yet mature enough to have attained commercial status; and most of the agent and service-oriented methods only focus on the analysis and design phases of the software development life cycle [23], [13]. This paper presents a novel agile tool kit (Java-based) to create, tailor and customize agile agent-oriented or agile service-oriented process fragments to create multiple abstraction paradigm-based (called here "m-abstraction") agile software development processes by using a method engineering approach [22]. In summary, we are currently developing an agile software solution framework, containing an embedded m-abstraction tool kit, which is being used and tested for the construction of agile processes for agent-oriented and service-oriented applications by using a method engineering approach.

This paper is organized as follows: Section 2 provides an overview of the required abstraction concepts. Section 3 describes agile and related concepts. Section 4 presents the organization of agile toolkit. Section 5 presents the evaluation and application of the agile toolkit with a case study. Finally, Section 6 presents the conclusions.

2 Abstraction: Agent and Service

A software system may be developed or modelled by using a number of abstraction mechanisms such as object, component, agent or service. An abstraction is a logical view of a real world problem or an entity from a specific (software) perspective; such as the representation of real world entities by objects, agents, services, components, features and procedures. These are all the examples of abstraction mechanisms.

2.1 Multi-abstraction or M-Abstraction (M-Oriented)

A software development project may combine more than one abstraction mechanism for a specific situation; for example, one project may involve the use of object-oriented and agent-oriented or agent-oriented and service-oriented abstraction together. In order to develop such a project, we need to have a methodology to support both abstractions at the same time. A method that combines practices to support more than one abstraction is called an m-abstraction or m-oriented method, whereas the project is called an m-abstraction or m-oriented project.

2.2 Characteristics of Agent Abstraction

An agent is an autonomous, interactive, communication-focused and flexible complex entity (existing in some environment) that cooperates with other agents to solve a complex software problem [18], [20], [36]. Indeed, the complex nature of agents makes multiagent systems difficult to build in comparison with object-oriented systems. Agents can be categorized as information attitudes and pro-attitudes. Information attitudes are: belief and knowledge. Pro-attitudes are: desire, intention, obligation, commitment and choice [5], [37]. A number of logical frameworks that