Chapter 2.7
Creating a Comprehensive Agent-Oriented Methodolgy:
Using Method Engineering and the OPEN Metamodel

Brian Henderson-Sellers
University of Technology, Sydney, Australia

ABSTRACT

While individual agent-oriented methodologies are useful for restricted situations, a more flexible approach can be found in the use of situational method engineering. Using an underpinning metamodel, a repository of method fragments can be built up and, from this, a selected number of fragments can be abstracted to form an organization-specific or project-specific methodology. As an example, we demonstrate the use of the OPEN metamodel and repository, as extended to support agent-oriented conceptual thinking. Re-creation of existing methodologies, such as Prometheus, is demonstrated with further enhancements from other methodologies such as Tropos and Gaia.

INTRODUCTION

Individual methodologies, such as those described so far in this book, are often created with specific purposes in mind, for example, particular domains, particular segments of the lifecycle. Used within those constraints, they are ideally suited. However, users often make the assumption that a methodology is not in fact constrained but, rather, is universally applicable. This can easily lead to "methodology failure" and the total rejection of methodological thinking by software development organizations (e.g., Avison & Fitzgerald, 2003).

The search for a one-size-fits-all methodology has been long and will, ultimately, always be fruitless (e.g., Cockburn, 2000). If the creation of a single universally applicable methodology is an unattainable goal, then we must ask how we might create a methodological environment in which the various demands of different software developers might be satisfied simultaneously. It is argued here that such flexibility might be best obtained by the use of method engineering, preferably based on an underlying metamodel. Formalizing methodological concepts and constraints with a metamodel in this way allows us to identify

two component parts of a methodology: (1) the process, and (2) various products created and/or used in the process (Rolland, Prakash, & Benjamen, 1999). Although all the chapters in this book deal with the broader term of "methodology" (i.e., process plus product), for our case study in this chapter, we will focus only on the "process" part of a methodology. A second assumption we make here is that an agent-oriented (AO) methodology can be formulated as a particular extension of an object-oriented (OO) methodology. In the context of method engineering and the earlier chapters of this book, we will demonstrate that this is a reasonable assumption.

In the following sections, we introduce situational method engineering as an effective approach for constructing a site-specific methodology that may be tailored or customized for individual projects in the context of OPEN, particularly as it has been partially extended to support agent-oriented software development. In the next section (CASE STUDY), we demonstrate how such methodology construction works with a simple case study and, finally, we conclude with a research and technology transfer agenda to facilitate the creation, finalization, and technology transfer of such a "standardized" approach in the context of the adoption of agent-oriented methodologies in an industrial context.

SITUATIONAL METHOD ENGINEERING

Method engineering (Brinkkemper, 1996; Kumar & Welke, 1992), more appropriately called situational method engineering or SME (Ter Hofstede & Verhoef, 1997), recognizes the fact (often ignored as noted above) that seeking to develop an all-encompassing methodology appropriate for all situations is foolish. Rather, SME seeks to model methodological processes and products by isolating conceptual method fragments (van Slooten & Hodes, 1996) that are coherent portions of a methodology. These fragments act as methodological "building blocks"—examples might be a particular technique to identify agent types or the way to use a sequence diagram in AUML (Odell, Van Dyke Parunak, & Bauer, 2000). These method fragments thus effectively allow a methodology to be extended by incorporating elements from other methodologies (e.g., Ralyté & Rolland, 2001). In other words, from these method fragments can be constructed a specialized methodology appropriate for each and every situation.

The methodologist publishes the method fragments, usually contained in a repository1 together with some construction guidelines (Brinkkemper, Saeki, & Harmsen Klooster, Brinkkemper, Harmsen, & Wijers, 1997; Ralyté & Rolland, 2001; Ralyté, Rolland, & Deneckère, 2004); and the user (usually an in-house method engineer) uses these construction guidelines to create the organization-specific or project-specific methodology within the organizational context and constraints (Figure 1). In the case where an organization develops software in a single domain, that constructed methodology may suffice for all developments; for other commercial developers, the constructed methodology may need to be project-specific.

Ideally, the elements in an SME repository should be compliant with (in fact, generated from) a set of concepts described by a metamodel (Henderson-Sellers, 2003; Ralyté & Rolland, 2001), which is a high-level description of the rules that govern the kinds of method fragments that are allowable. For instance, in the metamodel there may be an entity (often represented as a class) called Task.2 This definition describes the characteristics and rules of all tasks (for example, that they must be accomplished by a person using a special technique). From this Task concept in the metamodel can then be generated (usually by instantiation) very many kinds of tasks, for example, a task to create a class diagram, a task to identify the use cases, and so on. For each kind