A methodology is a body of methods employed by a discipline. A method is a procedure for attaining something. While some developers employ a handful of methods for developing agent-based systems, few have a methodology. Most developers of agent-based systems use an ad hoc approach—minimizing guidelines and assembling a seat-of-the-pants notation and terminology. While this approach provides maximum flexibility (as illustrated in Figure 1), the quality of the resulting application is questionable, since any knowledge and experience gained can not be easily transferred to other projects. Flexibility without control can hardly be considered a methodology, since any systematic and coordinated approach to establishing work methods is absent.

To be fair, choosing flexibility should not be considered a developer failing. Until recently, few methodologies existed from which to choose. This book is the first to present and explore the ten most prominent methodologies for developing agent-based systems. It is particularly valuable because readers will find suggested guidelines, techniques, notations, and terminology. Each of these methodologies contributes significantly toward aiding the developer.

Figure 1. Methodological approaches fall into three categories (adapted from Harmsen et al., 1994)
• The *Tropos* methodology provides guidance for the four major development phases of application development. One of its primary contributions is placing an emphasis on modeling goals and their relationship with the system’s actors, tasks, and resources.

• *MAS-CommonKADS* is based on both CommonKADS and object-oriented (OO)-based methodologies. This enables the developer to build agent-based systems while leveraging the experience of pre-agent methodologies and employing familiar techniques and diagrams.

• The *PASSI* methodology brings a particularly rich development lifecycle that spans initial requirements though deployment and, in addition, emphasizes the social model of agent-based systems.

• From an AI planning-execution perspective, *Prometheus* provides an especially rich goal-driven approach for its BDI-like agents. Its methodology is used today to develop systems on commercial BDI-based agent platforms, such as JACK or Agentis.

• *Gaia* is one of the earliest agent methodologies and now reflects this experience in Version Two of its approach. Using the analogy of human-based organizations, Gaia provides an approach that both a developer and a non-technical domain expert can understand—facilitating their interaction.

• *ADELFE* is a specialized methodology that emphasizes cooperative agents that self-organize and possibly result in emergent systems. More specifically, it addresses designing complex adaptive systems and anticipating emergence within its software agents.

• Resulting from a two-year, European-funded EURESCOM project, *MESSAGE* extends existing object-oriented methodologies for agent-oriented applications. Chartered to address telecommunications applications, its resulting RUP-based approach also supports more general applications.

• The *INGENIAS* methodology supports a notation based on five metamodels that define the different views and concepts of a multi-agent system. Using metamodels provides flexibility for evolving the methodology and adopting changes to its notation.

• *RAP* is concerned with distributed information systems (such as enterprise resource planning and supply-chain management systems) and places less emphasis on AI-based systems. The philosophy of the Model Driven Architecture (MDA) is adopted with the goal that executable software agents can be generated using RAP artifacts.

• *MaSE* is a comprehensive methodology that has been used to develop systems ranging from heterogeneous database integration applications to biologically based, computer-virus immune systems and cooperative robotics systems. Its hybrid approach can be applied to multi-agent sys-
tems that involve implementations, such as distributed human and machine planning.

In summary, all of the methodologies in this book provide valuable contributions that aid in the production of thoughtful agent-based applications. Each has its own unique perspective and approach to developing agent-based systems, while at the same time overlapping with many of the other approaches. Yet, no one methodology is useful in every system-development situation (the control approach depicted on the right of Figure 1).

This book presents the steps and notations for developing a variety of systems. By putting the methodologies side-by-side, the reader can select those portions relevant. Chapter 12 aids this process with a comprehensive overview and comparison of the various methodologies. Wisely, the approach adopted in the concluding chapter is controlled flexibility. Here, the most valuable building blocks—or method fragments—can be identified and extracted from each of the described methodologies and placed into a method repository. Such a repository will provide a broad set of reusable techniques that can be assembled based on the individual application’s profile.

By providing such a full range of methodologies, this book creates a balance between being evolutionary and revolutionary. On the one hand, the agent-based approach is not vastly different from current approaches. If a methodology differed radically from current practices, IT developers would experience difficulty in both understanding and accepting the approach. Furthermore, to suddenly reject the way we have developed applications for the last three decades would be unreasonable. Much of what we do as application developers does not change radically because the underlying software infrastructure has a high degree of stability. On the other hand, the agent-based approach to system development does differ from the conventional way of producing IT systems. Therefore, an agent methodology—even though it must still address and incorporate non-agent development requirements—is markedly different from traditional IT practices. Agents bring a new approach to developing systems. To one extent or another, the methodologies in this book involve identifying and incorporating those methodological features that make the agent-based approach possible while exploiting already-recognizable and time-tested processes. This is important for two reasons. Firstly, we need to communicate clearly to developers those steps necessary to create a successful agent-based implementation. Secondly, we must show why agent-based methodologies improve on traditional approaches—while still addressing those conventional aspects of an application.

In closing, I would like to say that I highly recommend this book as a guide to those developers and researchers involved in building agent-based applications.
It provides a wealth of guidelines, techniques, and notations for developing a variety of systems. Hence, I expect that its state-of-the-art approaches will heavily influence how the next generation of agent-based systems will be developed.

*James J. Odell*

**Reference**