Preface

Building software to solve contemporary business problems is no easy task. Over the last decade there has been an increasing focus on object-oriented notations and modelling languages, perhaps at the expense of a full methodological approach to solving the problem and giving software developers the tools they need to comprehensively create applications within management and market constraints—money, time, quality, and so forth. With increasingly sophisticated applications being demanded by businesses aiming for a competitive market advantage, object technologies are being supplemented and complemented by agent technologies. This is especially true in areas such as ambient intelligence, e-business, Web services, peer-to-peer networks, and bioinformatics. These areas demand software that is robust, can operate within a wide range of environments, can evolve over time to cope with changing requirements, is highly customizable to meet the needs of a wide range of users, and is sufficiently secure to protect personal data and other assets on behalf of its stakeholders. To fulfil these requirements, builders of systems need an appropriate *agent-oriented methodology*—the topic of this book.

Agent technology, increasing in popularity over the last decade, represents a concrete response to these new requirements. The main reason for this is that the agent paradigm, along with its definition of agent as an autonomous and proactive system capable of interacting with other agents in order to satisfy its objectives, provides a natural evolution for software systems. Agent-based systems call for new concepts, tools, and techniques for engineering and managing software. In particular, we need new software development methodologies that support the design and implement organizations of agents able to interact with one another in order to achieve some common or individual goal. However, in contrast to the history of object-oriented methodologies, in which industry played and is currently playing the major role, most of the agent-oriented methodologies are proposed by academic researchers and many of them are still in an early stage of maturity. Nevertheless, the time is ripe for an evaluation of the state-of-the art of agent-oriented methodologies, before they descend into the depths of a methodology jungle (as happened with object-oriented methodologies) that leads to industry rejection—spoiled for choice leads to "no choice" because it is unclear whether any of the individualistically proposed methodologies have any industrial future beyond the PhD scholarship or research grant supporting them in their (typically academic) research phase.

The intent of this book is therefore to give readers an understanding of the predominant and tested agent-oriented methodologies. The book characterizes each of these methodologies and compares them in terms of their main features. The book is organized as follows. Chapter I introduces what an agent-oriented methodology is, and it illustrates briefly the ten methodologies described in the rest of the book. Chapters II through XI, ably summarized by Jim J. Odell in the Foreword, then describe the methodologies, whereas Chapter XII presents an evaluation of all ten with a comparison of the methodologies based on the features-analysis approach. Finally, Chapter XIII illustrates how to create an agent-oriented methodology using method engineering based on the OPEN metamodel.

For each methodology, we asked the originators of that methodology to describe their work in the same way for each chapter. Within the space constraints we had given them (in order to maintain a balance across the book), we asked them to describe the current version of the methodology and then to illustrate this with a small case study. We also asked them to end their chapter with a short section to evaluate (from their viewpoint) what the strengths and weaknesses of their methodological approach are. After all, the authors are generally in the best position to know what critical issues they, and they alone, address – which is why they put the effort into creating the methodology in the first place. They are also in a good position to know the weaknesses, even if these are often hidden in more "marketing" presentations. Sometimes, omissions are purposeful yet seen by readers as "errors." So, we also asked the authors to state what they had omitted *and knew they had omitted*. When using abstraction techniques, which underpin methodologies as much as modelling, it is inevitable that some omissions and approximations will be used. Stating such constraints, say of restricted applicability to certain lifecycle stages or to certain classes of problems, makes the methodology even more valuable when applied in situations for which it has been designed.

We did not try to rationalize the notation. For many AO methodologies, UML or extensions thereof are selected. (We take the liberty of assuming that the reader is familiar with UML and therefore do not define it in this book.) Many in the community are exploring some more formal extensions to UML, such as Agent UML (AUML); others eschew these proposals and develop their own notation. Indeed, there is still some debate about whether extended an object-oriented notation like UML is the right way to go or whether the requirements of the agent community cannot be satisfied by waiting for the Object Management Group (OMG) to agree upon an agent-oriented extension to UML (although such efforts are indeed being discussed in the OMG in collaboration with the Foundation for Intelligent Physical Agents [FIPA]). We did ask that when authors used non-UML-like notations that they defined them in situ.

While the OO basis of many of these AO methodologies is evident (and discussed in more detail in Chapter I), it should be noted that a group of methodologies use the Rational Unified Process (RUP) as its basis. The terminology used in RUP, and therefore used in Chapters VII through IX, is not identical to the terminology used in the other chapters. Again, because of the different heritage we have not tried to rationalize these differences. We felt that had we done so, we would have destroyed the very essence of these three methodological approaches.

We are aware that much of the agent-oriented methodology development work, as reported here, has occurred in the realms of academe. Yet, one aim is to be able to transition these research results into industry. Particularly through our experiences in chairing various agent-oriented workshops (AOIS, AOSE, OOPSLA), we have become all too painfully aware of the challenge of transitioning agent technology into mainstream software development. The AO community is seen from outside as being undecided as to the basic characteristics of an agent, for example, whether or not to include proactivity and mobility and, if so, whether this is a binary characteristic or on a grey scale. From our previous experience with OO methodologies both in research and industry adoption, it is clear that there is a need for methodology developers to become aware of each others' work, to collaborate, to standardize, and to come up with a generally agreed approach to software development using their proposed new technology—here, agent technology. This book aims to be a first step in that direction. By presenting each methodology in as similar a light as possible and by undertaking a feature analysis, we can hope to gain insights, as researchers, into what works and what doesn't. As coalescence is encouraged, these ideas need to be tried out "for real" in industry applications. This is already beginning to happen, as is reported in some of the chapters in this book. Nevertheless, a concerted community effort is needed if industry adoption is to follow successfully.

In addition to presenting each methodology in an easy-to-understand way supplemented by an independent analysis (Chapter XII), we also propose a way forward for such a collaborative venture: situational method engineering. Chapter XIII offers insights into how this might work in the development of more flexible and coherent agent-oriented methodologies. The creation of a standard repository of method fragments, each of which captures the essence of some part of the methodology, allows industry adopters to "roll their own" methodology and thus establish an agreed in-house methodology specifically suited to their own peculiar circumstances. There are, to our knowledge, at least two international projects that are moving in this direction as well as significant research and standards development in the area of methodology metamodeling to underpin such a repository in a more formal way. Once established, it would be anticipated that the necessary tools would be built by third-party software developers/vendors. When all these pieces fall into place, we can then claim that agent-orientation is mainstream. At what date in the future this will occur, we hesitate to forecast; yet, we believe it is inevitable.

In closing, we wish to gratefully acknowledge the authors for their contributions and their patience in assisting us in putting together this book. We also owe them double thanks, since all chapters were reviewed by two people—the chapter authors again. In addition, we wish to thank, as valuable additional reviewers, John Debenham and Cesar Gonzalez-Perez of the University of Technology, Sydney. We trust that our joint effort will be a stimulus for industry in accepting and adopting the agent paradigm in the development of software systems.

Brian Henderson-Sellers, Sydney, Australia
Paolo Giorgini, Trento, Italy