Foreword to the 2nd Edition

In the young history of informatics, this book tells yet another story of connecting the world and the machine. Our discipline continuously attempts to create precise mathematical models of the world around us and of the basic mechanisms of our networked computer systems. These models have different properties that make them more or less appropriate to different goals. Their objective is not only to understand the word, but to help complement it and act on it. The main challenge is the alignment between the business system and the technical platform system. Many organizations struggle to meet their evolving business needs and goals in explicit and precise relation to their underlying technical information system. Shared abstract notations allow descriptions of both situations with common formalisms.

In the sixties and seventies, computer science pioneers proposed to bridge the problem space and the solution space through low-level constructs like procedures. Methods like top-down programming, or step-wise refinement, helped achieving this coupling. In the eighties, the object paradigm was found to be practical and efficient in describing the business side and the technical platform side at the same time. This has certainly triggered more work to study even richer abstractions based on different additional and combined paradigms like rules, relations, events, states, functions, services, and many more. Ways to accommodate multiparadigm systems have been investigated. Many tree-based notations have been proposed with XML technologies. The scope of this book, however, is that of graph-based notations like model-driven engineering or ontology engineering. One of the lessons it teaches is that there may not be a unique ideal abstraction to bridge the world problem space and the machine solution space. On the contrary, we may well have to live with different abstraction frameworks, different representation systems, and different technical or modeling spaces.

While there are some other books on similar subjects, this one is unique for several reasons. Instead of opposing different technologies or schools, it tries to understand them, to characterize them, to compare them, and finally to bridge them so that one may be able to use them in simultaneous or alternative ways when solving real problems. Technologies have to be agilely combined to contribute to solutions in a collaborative way. This book offers not only high-level conceptual presentations, but also implementation-level coverage of the presented technologies, and even hands-on guidance for their joint or separate applications to practical cases.

The authors take the reader through the entire presentation of the multiple facets of model-driven engineering and ontology engineering. They provide a wonderful pedagogical work that clearly and progressively introduces the main concepts, and their contribution may be recommended as an excellent introductory textbook on both technologies. They give an understanding of why and how these solutions may be concretely used in problem-solving and this itself is of tremendous interest to the researcher, the engineer, or the student that will read the book. Beyond these separate presentations, however, they relate them both conceptually and practically; and this is a complete originality of their contribution. The message is not about a silver bullet revealed, but instead about how different conceptual tools may be wisely selected and applied to achieve optimal solutions. New technologies are arriving on the market at a very rapid pace. It is hard to choose between them. Furthermore, as an organization accumulates assets in its information system, new technologies constantly emerge that seem to make previous ones obsolete. Technology interoperability must now be seriously considered. In this book, the authors have successfully performed a clever balancing act by producing a coherent and comprehensive guide on two technical spaces and a bridging framework that is well-grounded, both conceptually and practically. But the main message is that their method may also be generalized and applied to other technical spaces as well, and I am sure this will provide much inspiration for further work.

Finally, the reader will discover that the authors are presenting important variants of language engineering. Modeling languages and programming languages, general purpose languages, and domain-specific languages are becoming central to software engineering, to data engineering, and to system engineering. We know that the number of computer-based applications that will have to be built for various needs in upcoming decades is exponentially increasing. However, the number of professional computer scientists that will be available to produce these applications will follow a very slow linear progression. The only way out of this difficult situation is to mobilize computer scientists to not directly build the applications, but to provide the numerous domain languages that may guide end users to write precise and verifiable domain code themselves. At the end of this book, the reader will realize that she/he is now much more prepared to face these important new challenges of language engineering.

Nantes, France *Jean Bézivin*
February 2009