Preface

Introduction

Dilemmas involving notation, project planning, project management, and activity workflow pervade the world of software development. Object-orientation provides an elegant language for framing such problems, and powerful tools for resolving them.

In this book, we have brought together a collection of presentations, giving the reader an in-depth look into the technical, business, and social issues in managing object-oriented development processes, as well as presenting new technologies, making software development more effective. The chapters in the book examine many topics in the research frontier of software development, including methods, technologies, strategies, and the human factor. The book also presents the fundamentals of object-oriented project management.

The various backgrounds of the contributing authors—industrial, consulting, research, and teaching—yielded presentations, complementing and enriching each other. As a result, the book paints a holistic picture of the multi-faceted problems in object-oriented software development. It should be of interest to software developers, project managers, system analysts, and graduate and upper-level college students majoring in information systems and computer science who would like to deepen their knowledge in the field of object-oriented project management.

Very briefly, some of the major topics discussed in this book include: software development life cycle; development strategies, for example, open source, outsourcing, and product lines; componentization; the human factor; object-oriented notation and techniques, such as xUML, MDA, and MDSD; requirements engineering; design patterns; project management; and system integration with Web services.

Organization

The book is organized into 15 chapters. Each chapter emphasizes a particular area, identifies important shortcomings, discusses current activities, offers new insights into the problematics, and suggests opportunities for improving the management of object-oriented software development projects.

Motivated by computer simulation, the notions of object, class, and class generalization were formulated by Dahl and Nygaard in 1967. However, it was not until the mid-1990s that the first industrial-strength, object-oriented notations were complemented by sound development methods. Today, the object-oriented world is dominated by UML to the extent that UML and object-orientation have become synonymous. The book naturally begins with an introduction to UML2. The emphasis is on the novel features of UML and the new trends in object-orientation, namely, modeling of large things, a higher level of modeling abstraction, design automation, precision, and freedom from the constraints of the implementation platform.

In Chapter II, the themes from the introductory chapter are re-examined in the framework of xUML (executable UML) and MDA (model-driven architecture). MDA and xUML are among the latest initiatives of OMG. They promise to change the way software is created by combining a modeling language with a model manipulation language, rendering implementation programming obsolete. The chapter presents the two methodologies. It also discusses the MDA activity workflow and presents a development method for projects relying on xUML.

In Chapter III, Russ and McGregor present a model for planning object-oriented projects. The authors structure the software development landscape into a triad of high-level dimensions—technology, method, and organizational strategy—where each dimension is further divided into several sub-dimensions. The model defines project planning as navigating through a multi-dimensional hyperspace.

In Chapter IV, the Russ-McGregor model has been applied to evaluate the strength and weaknesses of xUML and MDA. The analysis sheds light on the economics of model-driven software development, and on the difficulties project managers and developers alike may encounter in adopting the two technologies in an industrial setting.

In Chapter V, Roussev and Akella present a new approach to managing outsourcing projects. Drawing on experience with Indian software firms, the authors closely analyze the problems faced by outsourcing clients and off-

shore developers. Roussev and Akella show how these problems can be successfully resolved by scaling down a large outsourcing project to meet the Agile "sweet spot," and by carefully managing the communication patterns among all stakeholders.

In Chapter VI, Roussev and Rousseva present a process extension applicable to both lightweight and heavyweight development methods. The extension is based on a business value invariant, and views the iterative and incremental model of software development as a communication model. The proposed techniques link the informal user requirements world to the system model, which makes it possible to derive mechanically the system architecture from the user requirements, and automatically to validate it with respect to the system's use case model through model animation.

It is a well-known fact that many of the agile practices are incompatible with the context of large-sized projects. Gary Pollice and Gary Evans, two nationally recognized methodologists, independently present their approaches to reproducing the conditions for agility in large-sized projects by balancing agility and discipline. Pollice and Evans look out for common grounds between Agile and RUP to get the best of both worlds.

In Chapter IX, Jorn Bettin, director of an international strategic technology management consultancy, addresses the question of how to create durable and scalable software architectures, so that the underlying design intent survives over a period of many years. Bettin goes beyond object-orientation and traditional iterative software development to define a set of guiding principles for component encapsulation and abstraction, and to form the foundation for a model-driven approach to software development.

In Chapter X, Magdy Serour from the Centre for Object Technology Applications and Research (COTAR) at the University of Technology, Sydney, delves into a gray area of object-orientation, namely, the effect of various human factors on the adoption and diffusion of an object-oriented software development process. Serour defines a process to assist organizations in planning and managing their transition to object-oriented development. The author discusses key "soft" factors, such as motivation, leadership, and overcoming the resistance to culture change, which are critical in promoting the process of organizational change.

In Chapter XI, Gerald Miller from Microsoft addresses a very important area of the new technological wave. Integration of systems in a cost-effective way is crucial for most enterprises, as many integration efforts fail to bring about the promised return on investment. Miller's presentation discusses how to

resolve the system integration nightmare by building a service-oriented architecture with Web services which integrates disparate systems, both within organizations and across business partners' firewalls.

In Chapter XII, de Lara, Guerra, and Vangheluwe give an overview of model-based software development, and propose ideas concerning meta-modeling and the use of visual languages for the specification of model transformations, model simulation, analysis, and code generation. They also examine the impact of model-based techniques on the development process.

The Agile methods are based on the presumption that a complete and stable requirements specification is generally impossible. This assumption invalidates the very vehicle for computing project velocity, progress, deadline prognosis, and budget allocations, as project managers cannot track the number of closed vs. open requirements. In Chapter XIII, Roock and Wolf demonstrate a practical technique, integrating lightweight mechanisms for project controlling into Agile methods. They propose to combining an (incomplete) hierarchical decomposition of a system with abstract measurements. Their approach addresses pressing management needs without incurring the burden of a waterfall-like exhaustive specification upfront.

Object-oriented knowledge comes in different forms, for example, principles, heuristics, patterns, refactoring, lessons learned, defects, and best practices. In Chapter XIV, Garzás and Piattini define an ontology of object-oriented micro-architectural design knowledge to systematize this knowledge so that it can be easily comprehended by developers and used in practical cases.

In the final chapter, Knott, Merunka, and Polak propose a new object-oriented methodology, which makes extensive use of business process modeling. The authors contrast and compare their approach to similar development approaches, and provide a case study to demonstrate the feasibility of their methodology.