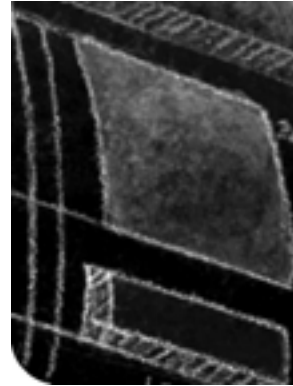


Preface



In teaching software engineering, experience has shown us that students are not convinced of the benefits of using software engineering techniques until they experience the benefits themselves. Completing a semester-long project is the most effective way of convincing students that software engineering is critical to their professional development. The software engineering course offered at Plymouth State College is therefore a very practical, hands-on course focused on the development of object-oriented software. Through the years, however, we became frustrated with the lack of textbooks appropriate for such a course. The majority of the available texts focus on the theoretical aspects of software engineering at the expense of its practical aspects. The texts that are project-based do not focus on the object-oriented paradigm. We wrote this textbook to fill this market gap.

This textbook focuses on actually performing software engineering. Theoretical concepts and terminology are introduced when they are necessary for successful software development. Although we recognize that there are a very large number of ways to develop software, we focus on a particular object-oriented software development methodology applied to a class project.

Having students engage in this semester-long team project also allows them to experience professional collaboration, which they seem to enjoy. Selecting an appropriate project is the most critical and most difficult aspect of teaching a project-based software engineering course. The project must be complex enough to engage a software development team of three to five students and yet be readily completed in fifteen weeks. More challenging than achieving proper scope is finding a project that interests and excites the students. To this end, we have provided a class project in the text. This project has been tested by Plymouth State College students and was successfully and enthusiastically implemented by a team of four students with varying programming and analytical skills.

This text is targeted to undergraduate computer science majors with little or no theoretical computer science in their background. The text is also written in a manner that is as programming language independent as possible. When language details are unavoidable, we have chosen Java as the programming language. We do not mean this text to be a reference manual of software engineering techniques and procedures. Instead, we provide a particular development methodology that will allow the completion of a significant software project over the course of a fifteen-week semester.

Since we assume the students will complete a project over the course of the semester, we have included the semester schedule for our course in Chapter 2. This schedule allows the project to begin swiftly at the start of the semester. Because we want students to experience as much of the software development methodology as possible, certain topics receive a less than comprehensive treatment. In particular, eliciting functional requirements from discussions with nontechnical users is a difficult task that requires much experience to accomplish successfully. Thus, we have presumed that the requirements will have been nearly completed by the instructor prior to the beginning of the semester.

To reinforce the practicality of the text, we also provide two running case studies. These are presented in a manner that models the development of the semester-long project. Sample deliverables are presented as part of the case studies to give students examples of the types of materials they are expected to deliver during the life cycle of their project.

Another important characteristic of this text is that it focuses on the object-oriented software development paradigm almost exclusively. Although we see the object-oriented approach as a logical extension of previous industry-adopted paradigms, this text is structured for an object-oriented project conceptualization, analysis, design, and implementation. A historical overview of software engineering techniques is presented to introduce students to the precursors to the object-oriented paradigm.

Although the long-enduring software crisis is not presented as the exclusive motivation for using software engineering techniques, a series of software development horror stories is included in the text so that students can see the results of ignoring various aspects of software engineering. Rather than addressing these stories as introductory material, they are included later in the text, so that there is less delay in getting to chapters needed to start the software engineering project.

In introducing the techniques that comprise the object-oriented paradigm, the Unified Modeling Language (UML) is used to model the software. Since UML is extremely large and intimidating, a subset of the notation is introduced on an “as-needed” basis. This book is not intended to serve as a comprehensive reference on UML. Many such references exist. Instead, UML is used as a tool in this text, much as it is used as a tool in the development of “real-world” software.

Pedagogical Features

- Each chapter begins with a list of important concepts that will be covered in the chapter.
- A class project that has been tested on our students runs throughout the text. The project is large enough for three to five students to complete over the course of a semester. Each chapter includes a set of activities that must be carried out in order to complete the class project. A specification of the deliverables for each part of the project is also included with the activities.
- Although the text includes a class project, the text is written so that an instructor can simply ignore the class project sections. If the instructor chooses to ignore the class project, a different project (or set of projects) can be substituted.
- Review questions are included at the end of each chapter. These exercises allow students additional practice with each of the topics covered in the book. They vary in complexity and difficulty.
- Exercises are included throughout each chapter. These exercises are most often presented as thought experiments and in most cases can be completed in class or out of class as deemed appropriate by the instructor.
- Unified Modeling Language is presented only when needed. When a particular modeling technique is needed for a particular step in the development methodology, the technique is described and examples are given. Through this approach, a subset of UML is presented.
- Two case studies run through the text. The first case study begins early in the text and is developed as the various steps in the methodology are presented. The second case study begins in Chapter 6, which acts as a review of the analysis and design phases of the development methodology. This second case study is then carried through the remainder of the text.
- Summary boxes are presented to allow a review of the development methodology at a quick glance.
- The chapters are organized so that students can realistically complete the class project in a single semester. For example, during the last four weeks of the class, while the students are engaged in coding and testing their projects and the topics of implementation and testing have already been covered, the text addresses topics such as project management, risk management, design patterns, and software development horror stories.
- Projects and schedules that have actually been tested in the classroom are included in the text.
- The last chapter of the text turns the tables on the students, requiring them to reflect on their experiences with the class project. It is entirely possible that the experience of some project development teams may be less successful than

others, so the discussion allows the students to review their course of action and suggest improvements. The final chapter guides students through a formal and professional presentation of their projects to the instructor and other classmates.

Supplements and Instructor Materials

Support materials are available to instructors adopting this textbook for classroom use and include the following:

- PowerPoint slides for each figure in the book
- PowerPoint lecture slides for each chapter
- Solutions for the *Questions for Review* sections
- Sample solutions or hints to spark discussion for the exercises embedded in each chapter
- A sample set of deliverables for the embedded class project
- Materials for two alternate class projects
- Source code for the Game2D case study.

Please check online information for this book at www.aw.com/cssupport for more information on obtaining these supplements.

In addition to these resources, we anticipate publishing a student supplement every two years that contains materials and exercises relevant for two different class projects.

Class Project

Instructors are encouraged to substitute their own projects, or an alternate project that has been provided as supplemental instructor material, for the specific class project included in the text. Each chapter that pertains to the development of the class project contains a section specifying class project-related goals and objectives. These sections have been written in a generic manner and should pertain to an alternate project as well as the project provided in the textbook.

There are a few sections of the book that address the included class project specifically. These sections have been included because the sample project serves as a particularly good example to illustrate a few of the design objectives discussed in the text. In order to understand these sections, students do not need to be familiar with the details of the included project, but rather simply need understand the idea of playing a multi-player game over the Internet. The examples address the sequence events comprising initiating such a multi-player game or making a board game-type move.

The following sections contain specific references to the class project:

- Section 1.10 contains the requirements specification. You may substitute an alternate project description here. This section should be skipped if an alternate project is being used.
- Section 2.2.2 illustrates a sample informal scenario, which can be easily understood by anyone familiar with common board games. We recommend using this example even if another class project is being used.
- Section 4.9 illustrates modeling the interprocess communication in the sample class project. This illustration can be easily understood by someone who is familiar with any multi-player, Internet-based game, so we recommend its use by those using an alternate class project.

Acknowledgments

Thanks to Maite Suarez-Rivas and Katherine Haruntunian from Addison Wesley for the incredibly positive support in the writing of this text. Jody Girouard was instrumental in the development of materials for instructors adopting the textbook as well as in providing feedback about the book from a student's perspective. John Girouard, Mark Henwood, Nick Rago, and David Sleeper were students in the first software engineering course that used the Use Case Centered Development methodology. They implemented *Galaxy Sleuth* and provided tremendous feedback about the methodology. Liz Johnson from Xavier University used an early version of the text in her software engineering class and helped us understand where our writing was unclear. Finally, numerous reviewers provided us with valuable feedback on the manuscript as it progressed. Some of these reviewers remain anonymous and we thank them for their work. The following reviewers examined later versions of the manuscript and helped us to tighten up our prose to provide better explanations:

Michael Beeson, San Jose State University
Jorge L. Diaz-Herrera, Southern Polytechnic State University
Jozo Dujmovic, San Francisco State University
Mohamed Fayad, University of Nebraska, Lincoln
J. W. Fendrich, Illinois State University
J. A. "Drew" Hamilton, Naval Postgraduate School
Alex Iskold, New York University
Jonathan Maletic, University of Memphis
Michael McCracken, Georgia Institute of Technology
Fatma Mili, Oakland University
Robert Noonan, College of William and Mary

Srini Ramaswamy, Tennessee Technological University

Steve Roach, University of Texas, El Paso

Don Shafer, Athens Group, Inc.

Bill Shay, University of Wisconsin, Green Bay

James E. Tomayko, Carnegie Mellon University

David Umphress, Auburn University

Shon Vick, University of Maryland, Baltimore County

Linda Werner, University of California, Santa Cruz

Janusz Zalewski, University of Central Florida