ENGINEERING SOFTWARE PRODUCTS

An Introduction to Modern Software Engineering

Ian Sommerville
Software products, such as stand-alone programs, web apps and services, and mobile apps, have transformed our everyday life and work. There are tens of thousands of software product companies, and hundreds of thousands of software engineers are employed worldwide in software product development.

Contrary to what some people may think, engineering software products needs more than coding skills. So, I’ve written this book to introduce some of the software engineering activities that are important for the production of reliable and secure software products.

**Who is the book for?**

The book has been designed for students taking a first course in software engineering. People thinking about developing a product who don’t have much software engineering experience may also find it useful.

**Why do we need a software engineering book that’s focused on software products?**

Most software engineering texts focus on *project-based* software engineering, where a client develops a specification and the software is developed by another company. However, the software engineering methods and techniques that have been developed for large-scale projects are not suited to software product development.
Students often find it difficult to relate to large, custom software systems. I think that students find it easier to understand software engineering techniques when they are relevant to the type of software that they constantly use. Also, many product engineering techniques are more directly relevant to student projects than project-oriented techniques.

Is this a new edition of your other software engineering textbook?

No, this book takes a completely different approach and, apart from a couple of diagrams, does not reuse any material from *Software Engineering*, 10th edition.

What’s in the book?

Ten chapters cover software products, agile software engineering, features, scenarios and user stories, software architecture, cloud-based software, microservices architecture, security and privacy, reliable programming, testing, and DevOps and code management.

I’ve designed the book so that it’s suitable for a one-semester software engineering course.

How is this book different from other introductory texts on software engineering?

As I said, the focus is on products rather than projects. I cover techniques that most other SE texts don’t cover, such as personas and scenarios, cloud computing, microservices, security, and DevOps. As product innovation doesn’t come from university research, there are no citations or references to research and the book is written in an informal style.
What do I need to know to get value from the book?

I assume that you have programming experience with a modern object-oriented programming language such as Java or Python and that you are familiar with good programming practice, such as the use of meaningful names. You should also understand basic computing concepts, such as objects, classes, and databases. The program examples in the book are written in Python, but they are understandable by anyone with programming experience.

What extra material is available to help teachers and instructors?

1. An instructor’s manual with solutions to exercises and quiz questions for all chapters
2. Suggestions how you can use the book in a one-semester software engineering course
3. Presentations for teaching (Keynote, PowerPoint, and PDF)

You can access this material at: https://www.pearsonhighered.com/sommerville
Additional material is available on the book’s website
https://iansommerville.com/engineering-software-products/

Where can I find out more?

I’ve written a couple of blog posts that are relevant to the book. These provide more information about my thoughts on teaching software engineering and my motivation for writing the book.

"Out with the UML (and other stuff too): reimagining introductory courses in software engineering"

"Engineering Software Products"
Acknowledgments

I’d like to thank the reviewers who made helpful and supportive suggestions when they reviewed the initial proposal for this book:

Paul Eggert—UCLA Los Angeles
Jeffrey Miller—University of Southern California
Harvey Siy—University of Nebraska Omaha
Edmund S. Yu—Syracuse University
Gregory Gay—University of South Carolina
Josh Delinger—Towson University
Rocky Slavin—University of Texas San Antonio
Bingyang Wei—Midwestern State University

Thanks also to Adam Barker from St. Andrews University for keeping me right on containers and to Rose Kernan who managed the production of the book.

Thanks, as ever, to my family for their help and support while I was writing the book. Particular thanks to my daughter Jane, who did a great job of reading and commenting on the text. She was a brutal editor! Her suggested changes significantly improved the quality of my prose.

Finally, special thanks to our newest family member, my beautiful grandson Cillian, who was born while I was writing this book. His bubbly personality and constant smiles were a very welcome distraction from the sometimes tedious job of book writing and editing.

Ian Sommerville
## Chapter 1  Software Products

1. The product vision  
2. Software product management  
3. Product prototyping  

### Key Points  

### Recommended Reading  

### Presentations, Videos, and Links  

### Exercises  

## Chapter 2  Agile Software Engineering

1. Agile methods  
2. Extreme Programming  
3. Scrum  

### Key Points  

### Recommended Reading  

### Presentations, Videos, and Links  

### Exercises  

<table>
<thead>
<tr>
<th>Chapter 1</th>
<th>Software Products</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>The product vision</td>
<td>7</td>
</tr>
<tr>
<td>1.2</td>
<td>Software product management</td>
<td>11</td>
</tr>
<tr>
<td>1.3</td>
<td>Product prototyping</td>
<td>16</td>
</tr>
<tr>
<td><strong>Key Points</strong></td>
<td></td>
<td>17</td>
</tr>
<tr>
<td><strong>Recommended Reading</strong></td>
<td></td>
<td>18</td>
</tr>
<tr>
<td><strong>Presentations, Videos, and Links</strong></td>
<td></td>
<td>18</td>
</tr>
<tr>
<td><strong>Exercises</strong></td>
<td></td>
<td>19</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 2</th>
<th>Agile Software Engineering</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Agile methods</td>
<td>20</td>
</tr>
<tr>
<td>2.2</td>
<td>Extreme Programming</td>
<td>24</td>
</tr>
<tr>
<td>2.3</td>
<td>Scrum</td>
<td>27</td>
</tr>
<tr>
<td><strong>Key Points</strong></td>
<td></td>
<td>47</td>
</tr>
<tr>
<td><strong>Recommended Reading</strong></td>
<td></td>
<td>48</td>
</tr>
<tr>
<td><strong>Presentations, Videos, and Links</strong></td>
<td></td>
<td>48</td>
</tr>
<tr>
<td><strong>Exercises</strong></td>
<td></td>
<td>49</td>
</tr>
</tbody>
</table>
# Contents

**Chapter 3**  Features, Scenarios, and Stories  
3.1  Personas  
3.2  Scenarios  
3.3  User stories  
3.4  Feature identification  
Key Points  
Recommended Reading  
Presentations, Videos, and Links  
Exercises  

**Chapter 4**  Software Architecture  
4.1  Why is architecture important?  
4.2  Architectural design  
4.3  System decomposition  
4.4  Distribution architecture  
4.5  Technology issues  
Key Points  
Recommended Reading  
Presentations, Videos, and Links  
Exercises  

**Chapter 5**  Cloud-Based Software  
5.1  Virtualization and containers  
5.2  Everything as a service  
5.3  Software as a service  
5.4  Multi-tenant and multi-instance systems  
5.5  Cloud software architecture  
Key Points  
Recommended Reading
Chapter 6  Microservices Architecture  150

6.1  Microservices  154
6.2  Microservices architecture  157
6.3  RESTful services  173
6.4  Service deployment  179
Key Points  182
Recommended Reading  183
Presentations, Videos, and Links  184
Exercises  184

Chapter 7  Security and Privacy  185

7.1  Attacks and defenses  188
7.2  Authentication  195
7.3  Authorization  201
7.4  Encryption  203
7.5  Privacy  213
Key Points  217
Recommended Reading  218
Presentations, Videos, and Links  219
Exercises  219

Chapter 8  Reliable Programming  221

8.1  Fault avoidance  223
8.2  Input validation  242
8.3  Failure management  249
Key Points  256