Quantitative Software Engineering Series

Description:
A benchmark text on software development and quantitative software engineering

"We all trust software. All too frequently, this trust is misplaced. Larry Bernstein has created and applied quantitative techniques to develop trustworthy software systems. He and C. M. Yuhas have organized this quantitative experience into a book of great value to make software trustworthy for all of us."

Barry Boehm

Trustworthy Systems Through Quantitative Software Engineering proposes a novel, reliability–driven software engineering approach, and discusses human factors in software engineering and how these affect team dynamics. This practical approach gives software engineering students and professionals a solid foundation in problem analysis, allowing them to meet customers' changing needs by tailoring their projects to meet specific challenges, and complete projects on schedule and within budget.

Specifically, it helps developers identify customer requirements, develop software designs, manage a software development team, and evaluate software products to customer specifications. Students learn "magic numbers of software engineering," rules of thumb that show how to simplify architecture, design, and implementation.

Case histories and exercises clearly present successful software engineers' experiences and illustrate potential problems, results, and trade–offs. Also featuring an accompanying Web site with additional and related material, Trustworthy Systems Through Quantitative Software Engineering is a hands–on, project–oriented resource for upper–level software and computer science students, engineers, professional developers, managers, and professionals involved in software engineering projects.

Contents:
Preface.

Acknowledgment.
PART 1: GETTING STARTED.

1. Think Like an Engineer Especially for Software.
1.1 Making a Judgment.
1.2 The Software Engineer's Responsibilities.
1.3 Ethics.
1.4 Software Development Processes.
1.5 Choosing a Process.
1.5.1 No–Method "Code and Fix" Approach.
1.5.2 Waterfall Method.
1.5.3 Spiral Method: Planned Risk Assessment–Driven Process.
1.5.4 Development Plan Approach.
1.5.5 Planned Incremental Development Process.
1.5.6 Agile Process, a Apparent Oxymoron.
1.6 Re-emergence of Model-Based Software Development.

1.7 Process Evolution.

1.8 Organization Structure.

1.9 Principles of Sound Organizations.

1.10 Short Projects—4 to 6 Weeks.

1.10.1 Project 1: Automating Library Overdue Book Notices.

1.10.2 Project 2: Ajax Transporters, Inc. Maintenance Project.

1.11 Problems.


2.1 People: Cultivate the Guru and Support the Majority.

2.1.1 How to Recognize a Guru.

2.1.2 How to Attract a Guru to Your Project.

2.1.3 How to Keep Your Gurus Working.

2.1.4 How to Support the Majority.

2.2 Product: "Buy Me!".

2.2.1 Reliable Software Products.

2.2.2 Useful Software Products.

2.2.3 Good User Experience.

2.3 Process: "OK, How Will We Build This?".

2.3.1 Agile Processes.

2.3.2 Object Oriented Opportunities.

2.3.3 Meaningful Metrics.

2.4 Project: Making It Work.

2.5 Problems.

2.6 Case Studies.

PART 2: ETHICS AND PROFESSIONALISM.

3. Software Requirements.

3.1 What Can Go Wrong With Requirements.

3.2 The Formal Processes.

3.3 Robust Requirements.

3.4 Requirements Synthesis.
3.5 Requirements Specification.
3.6 Quantitative Software Engineering Gates.
3.7 SQFD Technology.
3.8 ICED-T Metrics.
3.8.1 ICED-T Insights.
3.8.2 Using the ICED-T Model.
3.9 Development Sizing and Scheduling with Function Points.
3.9.1 Function Point Analysis Experience.
3.9.2 NCSLOC vs Function Points.
3.9.3 Computing Simplified Function Points (sFP).
3.10 Case Study: The Case of the No-Show Service.
3.11 Problems.

4. Prototyping.
4.1 Make It Work; Then Make It Work Right.
4.1.1 How to Get at the Governing Requirements.
4.1.2 Rapid Application Prototype.
4.1.3 What's Soft is Hard.
4.2 So What Happens Monday Morning?.
4.2.1 What Needs to Be Prototyped?.
4.2.2 How Do You Build a Prototype?.
4.2.3 How Is the Prototype Used?.
4.2.4 What Happens to the Prototype?.
4.3 It Works, But Will It Continue to Work?.
4.4 Case Study: The Case of the Driven Development.
4.4.1 Significant Results.
4.4.2 Lessons Learned.
4.4.3 Additional Business Histories.
4.5 Why is Prototyping So Important?.
4.6 Prototyping Deficiencies.
4.7 Iterative Prototyping.
4.8 Case Study: The Case of the Famished Fish.
4.9 Problems.
5. Architecture.

5.1 Architecture Is a System's DNA.

5.2 Pity the Poor System Administrator.

5.3 Software Architecture Experience.

5.4 Process and Model.

5.5 Components.

5.5.1 Components as COTS.

5.5.2 Encapsulation and Abstraction.

5.5.3 Ready or Not, Objects Are Here.

5.6 UNIX.

5.7 TL1.

5.7.1 Mission.

5.7.2 Comparative Analysis.

5.7.3 Message Formatting.

5.7.4 TL1 Message Formulation.

5.7.5 Industry Support of TL1.

5.8 Documenting the Architecture.

5.8.1 Diary or Log Document.

5.8.2 Debriefing Document.

5.8.3 Users of Architecture Documentation.

5.9 Architecture Reviews.

5.10 Middleware.

5.11 How Many Times Before We Learn?.

5.11.1 Comair Cancels 1,100 Flights on Christmas 2004.

5.11.2 Air Traffic Shutdown in September 2004.

5.11.3 NASA Misses Mars, 2004.

5.11.4 Case Study: The Case of the Preempted Priorities.

5.12 Financial Systems Architecture.

5.12.1 Typical Business Processes.

5.12.2 Product-Related Layer in the Architecture.

5.12.3 Finding Simple Components.
5.13 Design and Architectural Process.
5.14 Problems.

6. Estimation, Planning and Investment.
6.1 Software Size Estimation.
6.1.1 Pitfalls and Pratfalls.
6.1.2 Software Size Metrics.
6.2 Function Points.
6.2.1 Fundamentals of Function Point Analysis.
6.2.2 Brief History.
6.2.3 Objectives of Function Point Analysis.
6.2.4 Characteristics of Quality Function Point Analysis.
6.3 Five Major Elements of Function Point Counting.
6.3.1 External Input (EI).
6.3.2 External Output (EO).
6.3.3 External Inquiry EQ).
6.3.4 Internal Logical File (ILF).
6.3.5 External Interface Files (EIF).
6.4 Each Element Can Be Simple, Average or Complex.
6.5 Sizing an Automation Project with FPA.
6.5.1 Advantages of Function Point Measurement.
6.5.2 Disadvantages of Function Point Measurement.
6.5.3 Results Common to FPA.
6.5.4 FPA Accuracy.
6.6 SLOC Metric.
6.6.1 Company Statistics.
6.6.2 Reuse.
6.6.3 Wide Band Delphi.
6.6.4 Disadvantages of SLOC.
6.7 Production Planning.
6.7.1 Productivity.
6.7.2 Mediating Culture.
6.7.3 Customer Relations.
6.7.4 Centralized Support Functions.

6.8 Investment.


6.8.2 COCOMO.

6.8.3 Scheduling Tools—PERT, Gantt.

6.8.4 Project Manager's Job.

6.9 Problems.

7. Design for Trustworthiness.

7.1 Built-in Trustworthiness.

7.2 Software Reliability Overview.

7.3 Design Reviews.

7.3.1 Topics for the Design Review.

7.3.2 Case Study.

7.3.3 Interfaces.

7.3.4 Software Structure Influences Reliability.

7.3.5 Components.

7.3.6 Open & Closed Principle.

7.3.7 The Liskov Substitution Principle.

7.3.8 Comparing Object Oriented Programming with Componentry.

7.3.9 Politics of Reuse.

7.3.9.1 Qualified Successes.

7.3.9.2 Conditions Fostering Reuse.

7.3.9.3 Reuse "As Is".

7.4 Design Principles.

7.4.1 Strong Cohesion.

7.4.2 Weak Coupling.

7.4.3 Information Hiding.

7.4.4 Inheritance.

7.4.5 Generalization/Abstraction.

7.4.6 Separation of Concerns.

7.4.7 Removal of Context.
7.5 Documentation.
7.6 Design Constraints That Make Software Trustworthy.
7.6.1 Simplify the Design.
7.6.2 Software Fault Tolerance.
7.6.3 Software Rejuvenation.
7.6.4 Hire Good People and Keep Them.
7.6.5 Limit the Language Features Used.
7.6.6 Limit Module Size and Initialize Memory.
7.6.7 Check the Design Stability.
7.6.8 Bound the Execution Domain.
7.6.9 Have Performance Budgets and Engineer.
7.6.10 Reduce Algorithm Complexity.
7.6.11 Factor and Refactor.
7.7 Problems.

PART 3: TAKING THE MEASURE OF THE SYSTEM.
8. Identifying and Managing Risk.
8.1 Undesirable Events.
8.2 Risk Management Paradigm.
8.3 Functions of Risk Management.
8.4 Risk Analysis.
8.5 Calculating Risk.
8.6 Using Risk Assessment in Project Development: The Spiral Method.
8.7 Containing Risks.
8.7.1 Incomplete and Fuzzy Requirements.
8.7.2 Schedule Too Short.
8.7.3 Not Enough Staff.
8.7.4 Morale of Key Staff Is Poor.
8.7.5 Stakeholders Are Losing Interest.
8.7.6 Untrustworthy Design.
8.7.7 Feature Set Is Not Economically Viable.
8.7.8 Feature Set Is Too Large.
8.7.9 Technology Is Immature.
8.7.10 Late Planned Deliveries of Hardware and Operating System.

8.8 Manage the Cost Risk to Avoid Outsourcing.

8.8.1 Technology Selection.

8.8.2 Tools.

8.8.3 Software Manufacturing.

8.8.4 Integration, Reliability and Stress Testing.

8.8.5 Computer Facilities.

8.8.6 Human Interaction Design and Documentation.

8.9 Software Project Management Audits.

8.10 Running an Audit.

8.11 Risks with Risk Management.

8.12 Problems.


9.1 A Click in the Right Direction.

9.2 Managing Things, Managing People.

9.2.1 Knowledge Workers.

9.2.2 Collaborative Management.

9.3 FAA Rationale for Human Factors Design.

9.4 Reach Out and Touch Something.

9.5 System Effectiveness in Human Factors Terms.

9.5.1 What to Look for in COTS.

9.5.2 Simple Guidelines for Managing Development.

9.6 How Much Should the System Do?.

9.6.1 Screen Icon Design.

9.6.2 Short- and Long-Term Memory.

9.7 Emerging Technology.

9.8 Pleasing the Client by Pleasing the Developers.

9.9 The Bell Laboratories Philosophy.

9.10 So You Want to Be a Manager.

9.11 Problems.

10. Implementation Details.
10.1 Structured Programming.
10.2 Rational Unified Process and Unified Modeling Language.
10.3 Measuring Complexity.
10.4 Coding Styles.
10.4.1 Data Structures.
10.4.2 Team Coding.
10.4.3 Code Reading.
10.4.4 Code Review.
10.4.5 Code Inspections.
10.5 A Must Read for Trustworth Software Engineers.
10.6 Coding for Parallelism.
10.7 Threats.
10.8 Open Source Software.
10.9 Problems.

11.1 The Price of Quality.
11.1.1 Unit Testing.
11.1.2 Integration Testing.
11.1.3 System Testing.
11.1.4 Reliability Testing.
11.1.5 Stress Testing.
11.2 Robust Testing.
11.2.1 Robust Design.
11.2.2 Prototypes.
11.2.3 Identify Expected Results.
11.2.4 Orthogonal Array Test Sets (OATS).
11.3 Testing Techniques.
11.3.1 One–Factor–at–a–Time.
11.3.2 Exhaustive.
11.3.3 Deductive Analytical Method.
11.3.4 Random/Intuitive Method.
11.3.5 Orthogonal Array–Based.
11.3.6 Defect Analysis.
11.4 Case Study: Web Time Charging System (TCS).
11.5 Cooperative Testing.
11.6 Graphic Footprint.
11.7 Testing Strategy.
11.7.1 Test Incrementally.
11.7.2 Test Under No Load.
11.7.3 Test Under Medium Load.
11.7.4 Test Under Heavy Load.
11.7.5 Test Under Overload.
11.7.6 Test the Error Recovery Code.
11.7.7 Diabolic Testing.
11.7.8 Reliability Tests.
11.7.9 Footprint.
11.7.10.
11.7.11 Regression.
11.8 Software Hot Spots.
11.9 Software Manufacturing Defined.
11.10 Configuration Management.
11.11 Outsourcing.
11.11.1 Test Modules.
11.11.2 Faster Iteration.
11.11.3 Meaningful Test Process Metrics.
11.12 Problems.
12. The Final Project: By Students, For Students.
12.1 How to Make the Course Work for You.
12.2 Sample Call for Projects.
12.3 A Real Student Project.
12.4 The Rest of the Story.
12.5 Our Hope.
Index.
Fax Order Form
To place an order via fax simply print this form, fill in the information below and fax the completed form to 646-607-1907 (from USA) or +353-1-481-1716 (from Rest of World). If you have any questions please visit http://www.researchandmarkets.com/contact/

Order Information
Please verify that the product information is correct.

Product Name: Trustworthy Systems Through Quantitative Software Engineering, Quantitative Software Engineering Series
Web Address: http://www.researchandmarkets.com/reports/2173091/
Office Code: SCD2LHLA

Product Format
Please select the product format and quantity you require:

<table>
<thead>
<tr>
<th>Quantity</th>
<th>USD 138 + USD 29 Shipping/Handling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hard Copy (Hard Back):</td>
<td></td>
</tr>
</tbody>
</table>

* Shipping/Handling is only charged once per order.

Contact Information
Please enter all the information below in BLOCK CAPITALS

Title: Mr [ ] Mrs [ ] Dr [ ] Miss [ ] Ms [ ] Prof [ ]
First Name: ____________________________ Last Name: ____________________________
Email Address: * ____________________________
Job Title: ____________________________
Organisation: ____________________________
Address: ____________________________
City: ____________________________
Postal / Zip Code: ____________________________
Country: ____________________________
Phone Number: ____________________________
Fax Number: ____________________________

* Please refrain from using free email accounts when ordering (e.g. Yahoo, Hotmail, AOL)
Payment Information

Please indicate the payment method you would like to use by selecting the appropriate box.

☐ Pay by credit card: You will receive an email with a link to a secure webpage to enter your credit card details.

☐ Pay by check: Please post the check, accompanied by this form, to:
Research and Markets,
Guinness Center,
Taylors Lane,
Dublin 8,
Ireland.

☐ Pay by wire transfer: Please transfer funds to:

<table>
<thead>
<tr>
<th>Account number</th>
<th>Sort code</th>
<th>Swift code</th>
<th>IBAN number</th>
<th>Bank Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>833 130 83</td>
<td>98-53-30</td>
<td>ULSBIE2D</td>
<td>IE78ULSB98533083313083</td>
<td>Ulster Bank, 27-35 Main Street, Blackrock, Co. Dublin, Ireland.</td>
</tr>
</tbody>
</table>

If you have a Marketing Code please enter it below:

Marketing Code: _______________________________________

Please note that by ordering from Research and Markets you are agreeing to our Terms and Conditions at http://www.researchandmarkets.com/info/terms.asp

Please fax this form to:
(646) 607-1907 or (646) 964-6609 - From USA
+353-1-481-1716 or +353-1-653-1571 - From Rest of World