1. Record Nr. UNINA990008269210403321

| | |
|---|---|
| Autore | Prunier, Fernand |
| Titolo | Sur le théorème de Fermat : Fermat l'a t-il démontré? / Fernand Prunier |
| Pubbl/distr/stampa | Paris : Librairie scientifique et technique Albert Blanchard, 1966 |
| Descrizione fisica | 9 p. ; 27 cm |
| Locazione | FI1 |
| Collocazione | 4A-192 |
| Lingua di pubblicazione | Francese |
| Formato | Materiale a stampa |
| Livello bibliografico | Monografia |

2. Record Nr. UNINA9910814546703321

| | |
|---|---|
| Autore | Lui Kim Man |
| Titolo | Software development rhythms : harmonizing agile practices for synergy / / Kim Man Lui and Keith C.C. Chan |
| Pubbl/distr/stampa | Hoboken, N.J., : Wiley-Interscience, c2008 |
| ISBN | 9786611373900<br>9781281373908<br>1281373907<br>9780470192672<br>0470192674<br>9780470192665<br>0470192666 |
| Edizione | [1st ed.] |
| Descrizione fisica | 1 online resource (325 p.) |
| Altri autori (Persone) | ChanKeith C. C |
| Disciplina | 005.1 |
| Soggetti | Computer software - Development |
| Lingua di pubblicazione | Inglese |
| Formato | Materiale a stampa |
| Livello bibliografico | Monografia |
| Note generali | Description based upon print version of record. |
| Nota di bibliografia | Includes bibliographical references and index. |

Nota di contenuto

SOFTWARE DEVELOPMENT RHYTHMS; CONTENTS; PREFACE; Special Acknowledgment; Part I: Essentials; 1 NO PROGRAMMER DIES; 1.1 Developing Software versus Building a Tunnel; 1.1.1 The Good Old Days?; 1.1.2 The More Things Change, the More They Stay the Same?; 1.1.3 Behind Software Products; 1.1.4 Deal or No Deal; 1.2 Do-Re-Mi Do-Re-Mi; 1.2.1 Iterative Models; 1.2.2 Code and Fix; 1.2.3 Chaos; 1.2.4 Methodology that Matters; 1.3 Software Development Rhythms; 1.3.1 Stave Chart by Example; 1.3.2 Game Theory; 1.3.3 In-Out Diagram; 1.3.4 Master-Coach Diagram; 1.3.5 No Mathematics
1.3.6 Where to Explore RhythmsReferences; 2 UNDERSTANDING PROGRAMMERS; 2.1 Personality and Intelligence; 2.1.1 Virtuosi; 2.1.2 Meeting Your Team; 2.1.3 Recruiting Programmers; 2.2 Outsourced Programmers; 2.2.1 Programmers in Their Environments; 2.2.2 Programmers, Cultures, and Teams; 2.3 Experienced Management; 2.3.1 Being Casual about Causal Relationships; 2.3.2 Not Learning from Experience; 2.3.3 Doing Things Right Right Now; References; 3 START WITH OPEN SOURCE; 3.1 Process and Practice; 3.1.1 The Four Ps of Projects; 3.1.2 Agile Values; 3.1.3 Zero-Point Collaboration
3.2 Open-Source Software (OSS) Development3.2.1 Software Cloning; 3.2.2 Software Quality; 3.2.3 Starting Processes; 3.2.4 Open-Source Development Community; 3.2.5 Ugrammers; 3.2.6 Participant Roles; 3.2.7 Rapid Release; 3.2.8 Blackbox Programming; 3.2.9 OSS Practices; 3.3 OSS-Like Development; 3.3.1 Agile Practices; 3.3.2 Communication Proximity; 3.3.3 Loose and Tight Couples; 3.3.4 Collocated Software Development; 3.4 Conclusion; References; Part II: Rhythms; 4 PLAGIARISM PROGRAMMING; 4.1 Plagiarism; 4.1.1 Existing Code; 4.1.2 Social Network Analysis; 4.1.3 Being Plagiarized
4.1.4 Turn Everyone into a Programmer4.1.5 Pattern Language; 4.1.6 Software Team Capability; 4.1.7 Rough-Cut Design; 4.1.8 Training Is Not a Solution; 4.2 Nothing Faster than Plagiarism; 4.2.1 Immorality; 4.2.2 Unprecedented Code; 4.2.3 People Network; 4.2.4 Rhythm for Plagiarism; 4.2.5 Plagiarism at Work; 4.3 Business and Rhythm for Plagiarism; 4.3.1 15-Minute Business Presentation; 4.3.2 Marketing Research; 4.3.3 Chatting Robot; 4.3.4 Old Song, New Singer; References; 5 PAIR PROGRAMMING; 5.1 Art and Science; 5.1.1 The Right Partner; 5.1.2 Noisy Programming; 5.1.3 Just Training
5.1.4 Pay to Watch5.2 Two Worlds; 5.2.1 Moneyless World; 5.2.2 Money-Led World; 5.2.3 Economics; 5.2.4 Mythical Quality-Time; 5.2.5 Elapsed Time Accelerated; 5.2.6 Critical Path Method; 5.2.7 Why Two, Not Three: The Antigroup Phenomenon; 5.2.8 Software Requirements Are Puzzles; 5.3 Programming Task Demands; 5.3.1 2 and 4 Is 6; 5.3.2 2 and 4 Is 4; 5.3.3 2 and 4 Is 3; 5.3.4 2 and 4 2; 5.3.5 2 and 4 is Unknown; 5.4 Pair Programming Is More than Programming; 5.4.1 Design by Code; 5.4.2 Pair Design; 5.4.3 Rhythmic Pair Programming; 5.5 Pair Programming Team Coached; References
6 REPEAT PROGRAMMING

Sommario/riassunto

An accessible, innovative perspective on using the flexibility of agile practices to increase software quality and profitability When agile approaches in your organization don't work as expected or you feel caught in the choice between agility and discipline, it is time to stop and think about software development rhythms! Agile software development is a popular development process that continues to reshape philosophies on the connections between disciplined processes and agile practices. In Software Development Rhythms, authors Lui and Chan explain how adopting one practice and combining