Software craftsmanship is a call to arms for professional software developers to master the skills they need to succeed in their role. Software craftsmanship emphasizes the coding skills of the software developers and is a unique approach that emphasizes the importance of developer accountability.

Historically, programmers have been encouraged to see themselves as practitioners of the well-defined disciplines such as agile and lean, taking all their development projects to the next level. By placing greater emphasis on the mindset required for both developers and their employers, software craftsmanship helps improve upon best-practice technical disciplines to achieve excellence.

The right mindset - for both developers and their employers. As developers worldwide are recognizing, the right mindset is craftsmanship...

This book introduces the author's collection of wisdom under one umbrella: Software Craftsmanship. This approach is unique in that it spells out a programmer-centric way to build software. In other words, all the best computers, proven components, and most robust languages mean nothing if the programmer does not understand their craft.

This book can help you make the best of both your life and your career. Solutions to some common obstacles that this book explores in depth include: Burned out at work? "Nurture Your Passion" by finding a pet project to rediscover the joy of problem solving. Feeling overwhelmed and insecure? "Be the Worst" for a while. And when you're ready for the next level, "Be the Best" toward financial success.

Are you doing all you can to further your career as a software developer? With today's rapidly changing and ever-expanding technologies, being successful requires more than technical expertise. To grow professionally, you also need soft skills and effective learning techniques.

Software craftsmanship is an approach to software development that emphasizes the coding skills of the software developers. It is a response by software developers to the perceived ills of the mainstream software industry, including the prioritization of financial concerns over developer accountability. Historically, programmers have been encouraged to see themselves as practitioners of the well-defined disciplines such as agile and lean, taking all their development projects to the next level.

As developers worldwide are recognizing, the right mindset is craftsmanship...
Acces PDF Software Craftsmanship The New Imperative

patterns and test-driven development—focuses on these critical decisions, unearthing powerful “implementation patterns” for writing programs that are simpler, clearer, better organized, and more cost effective. Beck collects 77 patterns for handling everyday programming tasks and writing more readable code. This new collection of patterns addresses many aspects of development, including class, state, behavior, method, collections, frameworks, and more. He uses diagrams, stories, examples, and essays to engage the reader as he illuminates the patterns. You’ll find proven solutions for handling everything from naming variables to checking exceptions.

Widely considered one of the best practical guides to programming, Steve McConnell’s original CODE COMPLETE has been helping developers write better software for more than a decade. Now this classic book has been fully updated and revised with leading-edge practices—and hundreds of new code samples—illustrating the art and science of software construction. Capturing the body of knowledge available from research, academia, and everyday commercial practice, McConnell synthesizes the most effective techniques and must-know principles into clear, pragmatic guidance. No matter what your experience level, development environment, or project size, this book will inform and stimulate your thinking—and help you build the highest quality code. Discover the timeless techniques and strategies that help you:

- Design for minimum complexity and maximum creativity
- Reap the benefits of collaborative development
- Apply defensive programming techniques to reduce and flush out errors
- Exploit opportunities to refactor—or evolve—code, and do it safely
- Use construction practices that are right-weight for your project
- Debug problems quickly and effectively
- Resolve critical construction issues early and correctly
- Build quality into the beginning, middle, and end of your project

Ever since Extreme Programming burst on to the application development scene in 1998, it has been a lightning rod for controversy. In “Questioning Extreme Programming,” author McBreen puts this agile approach to application development under the microscope, and closely examines both sides of this heated debate.

Vaughn Vernon presents concrete and realistic domain-driven design (DDD) techniques through examples from familiar domains, such as a Scrum-based project management application that integrates with a collaboration suite and security provider. Each principle is backed up by realistic Java examples, and all content is tied together by a single case study of a company charged with delivering a set of advanced software systems with DDD.

CD-ROM contains cross-referenced code.

Peter Seibel interviews 15 of the most interesting computer programmers alive today in Coders at Work, offering a companion volume to Apress’s highly acclaimed best-seller Founders at Work by Jessica Livingston. As the words “at work” suggest, Peter Seibel focuses on how his interviewees tackle the day-to-day work of programming, while revealing much more, like how they became great programmers, how they recognize programming talent in others, and what kinds of problems they find most interesting. Hundreds of people have suggested names of programmers to interview on the Coders at Work web site: www.codersatwork.com. The complete list was 284 names.

Having digested everyone’s feedback, we selected 15 folks who’ve been kind enough to agree to be interviewed: Frances Allen: Pioneer in optimizing compilers, first woman to win the Turing Award (2006) and first female IBM fellow

Joe Armstrong: Inventor of Erlang

Joshua Bloch: Author of the Java collections framework, now at Google

Bernie Cosell: One of the main software guys behind the original ARPANET IMPs and a master debugger

Douglas Crockford: JSON founder, JavaScript architect at Yahoo!

L. Peter Deutsch: Author of Ghostscript, implementer of Smalltalk-80 at Xerox PARC and Lisp 1.5 on PDP-1

Brendan Eich: Inventor of JavaScript, CTO of the Mozilla Corporation

Brad Fitzpatrick: Writer of LiveJournal, OpenID, memcached, and Perlbal

Dan Ingalls: Smalltalk implementor and designer

Simon Peyton Jones: Coinventor of Haskell and lead designer of Glasgow Haskell Compiler

Donald Knuth: Author of The Art of Computer Programming

Peter Norvig: Director of Research at Google and author of the standard text on AI

Guy Steele: Coinventor of Scheme and part of the Common Lisp Gang of Five, currently working on Fortress

Ken Thompson: Inventor of UNIX

Jamie Zawinski: Author of XEmacs and early Netscape/Mozilla hacker