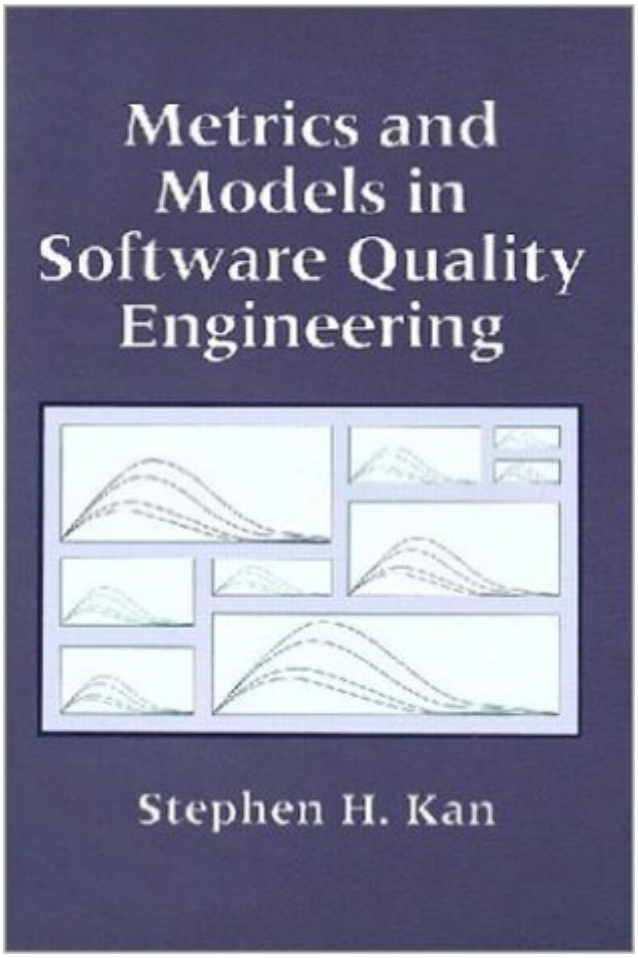
Metrics And Models In Software Quality Engineering



Synopsis

If you need to understand how to measure software quality and how to use measurements to improve your software development, you will want to have a copy of this book. Metrics and Models in Software Quality Engineering provides the information and teaches the skills you need to measure and improve the quality of the entire software development process from high-level to low-level design, as well as all phases of reliability. Joining action plans with actual project experiences, this book focuses on using - not just describing - metrics. It provides detailed coverage of essential issues and techniques, including software metrics, software reliability models, and models and analysis of program complexity. Metrics and Models in Software Quality Engineering goes even further, discussing such topics as in-process metrics, defect removal effectiveness, customer satisfaction, and more. Numerous real-life examples, many taken from the author’s experience as the software quality focal point for IBM’s Baldrige Award-winning AS/400, show you how to put the theories and techniques to work. The book also contains examples from such major computer companies as Hewlett-Packard, Motorola, and the NASA Software Engineering Laboratory. This excellent balance of theory, techniques, and examples makes for a highly-instructive and practical book on one of the most important topics in software development. "I’ve devoted considerable space to Kan’s Metrics and Models in Software Quality Engineering because I believe it is an important book that bridges the worlds of industrial statistical process control and software engineering. The AS/400 software is a large, complex and very successful product for IBM. Kan provides insights into the methods IBM used to control the quality in this project which provide lessons that we would all do well to study." -Software Development "The concise and clear explanation of function point counting is a jewel. Metrics and Models in Software Quality Engineering If you are looking for just one book on metrics, this is a good choice." -The Northwest C++ Users Group newsletter 0201633396B04062001

Book Information

Hardcover: 344 pages
Publisher: Addison-Wesley Professional; 1st edition (December 1994)
Language: English
ISBN-10: 0201633396
ISBN-13: 978-0201633399
Product Dimensions: 9.3 x 6.4 x 0.6 inches
Shipping Weight: 2.1 pounds

Average Customer Review: 4.4 out of 5 starsÂ Â See all reviewsÂ (17 customer reviews)
Best Sellers Rank: #584,103 in Books (See Top 100 in Books) #15 inÂ Books > Computers & Technology > Programming > Software Design, Testing & Engineering > Quality Control #258 inÂ Books > Business & Money > Management & Leadership > Quality Control & Management > Total Quality Management #920 inÂ Books > Computers & Technology > Business Technology > Software > Enterprise Applications

Customer Reviews

As the System Test Team Leader for the Quality Technology area, I had to certify many of the tools and procedures used by Stephen Kan. Prior to that, as a System Administrator, I had to run software metrics on those tools, like those shown in table 6.3. As a design review moderator, I was charged with leading a number of IR, IE and I0 reviews.THIS STUFF WORKS!I can attest personally to the great effort and many find minds that worked together to develop and implement fine tools such as DCR, PTR, PTF and APAR, as well as the brilliantly simple, effective ways of implementing Continuous Quality Improvement techniques such as DPP. What Kan has written is real-world honest and true, not some academic exercise.Kan is dead-right on the money. If you want to track, predict and manage things in the real world, this is how you do it. At PSQT ’97, Tom Gilb told me that SEI should create a new CMM "Level Six" designation for the way Kan and the others at IBM Rochester have dealt with software quality. That’s how good the stuff in this book is.I am particularly impressed by how Kan has woven in not only his work and IBM Rochester as a whole, but also the work of others throughout the industry in such a simple, clear, easy to understand manner.Yet, given that the book is an easy read, that many of the techniques are easy to do, and that I see this book on the shelves of many IT managers, it baffles me why so few people and so few companies actually implement this stuff. I suspect that politics and corporate culture is what’s holding back so many companies from enjoying the success, efficiency, and frankly the FUN of working in a continually measuring, continually refining work environment such as Kan describes.

The book has a metric for SW failure frequency. So the author takes a "Rayleigh distribution" (Weibull continuous distribution that can be used only when the random variable is probability, not a discrete number) - the shape parameter is =2 and the scale parameter is equal to tmax (meaning the fit is guaranteed), multiplies this with the number of initial faults (where does thin come from) and says that this cumulative (CDF) is the number of faults discovered. Weibull distribution is probabilistic and does not count so it CACCOT produce or represent any numbers. The CDF is

probability. Then the great math wizard takes a first derivative of this CDF and claims that this result is the failure frequency. This author is not the first mathematically ignorant person who confued the Weibull distribution for the Weibull Intensity Function (which is the curve fit for the non-homogenous Poisson distribution power law curve, and where the number of failures in time ARE fitted with this curve which looks like Weibull distribution but it is NOT Weibull, It is a Poisson process and Poisson DISCRETE distribution where the first derivative of failure fitting curve IS the failure frequency. Sadly, maybe hundreds of people as mathematically and physically ignorant, use this nonsense and get nonsense results, where PDF multiplied by the guessed INITIAL number of faults is named to be failure frequency. Another ignorance is claiming that this is software property. If the author has even seen software design he should know that the software test is for finding faults tested by people for faults. So, this "frequency" is the people frequency and not software failure frequency.

Dmca