

Creating A Software Engineering Culture

What company doesn't want energized workers, delighted customers, genuine efficiency, and breakthrough innovation? The Lean Mindset shows how lean companies really work-and how a lean mindset is the key to creating stunning products and delivering amazing services. Through cutting-edge research and case studies from leading organizations, including Spotify, Ericsson, Intuit, GE Healthcare, Pixar, CareerBuilder, and Intel, you'll discover proven patterns for developing that mindset. You'll see how to cultivate product teams that act like successful startups, create the kind of efficiency that attracts customers, and leverage the talents of bright, creative people. The Poppendiecks weave lean principles throughout this book, just as those principles must be woven throughout the fabric of your truly lean organization. Learn How To Start with an inspiring purpose, and overcome the curse of short-term thinking Energize teams by providing well-framed challenges, larger purposes, and a direct line of sight between their work and the achievement of those purposes Delight customers by gaining unprecedented insight into their real needs, and building products and services that fully anticipate those needs Achieve authentic, sustainable efficiency without layoffs, rock-bottom cost focus, or totalitarian work systems

Develop breakthrough innovations by moving beyond predictability to experimentation, beyond globalization to decentralization, beyond productivity to impact. Lean approaches to software development have moved from novelty to widespread use, in large part due to the principles taught by Mary and Tom Poppendieck in their pioneering books. Now, in The Lean Mindset, the Poppendiecks take the next step, looking at a company where multidiscipline teams are expected to ask the right questions, solve the right problems, and deliver solutions that customers love. In the course of their 20+-year engineering careers, authors Brian Fitzpatrick and Ben Collins-Sussman have picked up a treasure trove of wisdom and anecdotes about how successful teams work together. Their conclusion? Even among people who have spent decades learning the technical side of their jobs, most haven't really focused on the human component. Learning to collaborate is just as important to success. If you invest in the "soft skills" of your job, you can have a much greater impact for the same amount of effort. The authors share their insights on how to lead a team effectively, navigate an organization, and build a healthy relationship with the users of your software. This is valuable information from two respected software engineers whose popular series of talks—including "Working with Poisonous People"—has attracted hundreds of thousands of followers.

Introducing The Effective Engineer--the only

book designed specifically for today's software engineers, based on extensive interviews with engineering leaders at top tech companies, and packed with hundreds of techniques to accelerate your career.

A new edition of a graduate-level machine learning textbook that focuses on the analysis and theory of algorithms. This book is a general introduction to machine learning that can serve as a textbook for graduate students and a reference for researchers. It covers fundamental modern topics in machine learning while providing the theoretical basis and conceptual tools needed for the discussion and justification of algorithms. It also describes several key aspects of the application of these algorithms. The authors aim to present novel theoretical tools and concepts while giving concise proofs even for relatively advanced topics. Foundations of Machine Learning is unique in its focus on the analysis and theory of algorithms. The first four chapters lay the theoretical foundation for what follows; subsequent chapters are mostly self-contained. Topics covered include the Probably Approximately Correct (PAC) learning framework; generalization bounds based on Rademacher complexity and VC-dimension; Support Vector Machines (SVMs); kernel methods; boosting; on-line learning; multi-class classification; ranking; regression; algorithmic stability; dimensionality reduction; learning automata and languages; and reinforcement learning. Each chapter ends with a set of

exercises. Appendixes provide additional material including concise probability review. This second edition offers three new chapters, on model selection, maximum entropy models, and conditional entropy models. New material in the appendixes includes a major section on Fenchel duality, expanded coverage of concentration inequalities, and an entirely new entry on information theory. More than half of the exercises are new to this edition.

How to Navigate Clueless Colleagues, Lunch-Stealing Bosses, and the Rest of Your Life at Work

**Learning to Thrive with Self-Managing Teams
Team Geek**

**Implementing Lean Software Development
Accelerate**

**Lessons Learned from Programming Over Time
Creating a Software Engineering Culture**

Today, software engineers need to know not only how to program effectively but also how to develop proper engineering practices to make their codebase sustainable and healthy. This book emphasizes this difference between programming and software engineering. How can software engineers manage a living codebase that evolves and responds to changing requirements and demands over the length of its life? Based on their experience at Google,

File Type PDF Creating A Software Engineering Culture

software engineers Titus Winters and Hyrum Wright, along with technical writer Tom Manshreck, present a candid and insightful look at how some of the world's leading practitioners construct and maintain software. This book covers Google's unique engineering culture, processes, and tools and how these aspects contribute to the effectiveness of an engineering organization. You'll explore three fundamental principles that software organizations should keep in mind when designing, architecting, writing, and maintaining code: How time affects the sustainability of software and how to make your code resilient over time How scale affects the viability of software practices within an engineering organization What trade-offs a typical engineer needs to make when evaluating design and development decisions

Winner of the Shingo Publication Award Accelerate your organization to win in the marketplace. How can we apply technology to drive business value? For years, we've been told that the performance of software delivery teams doesn't matter?that it can't provide a

File Type PDF Creating A Software Engineering Culture

competitive advantage to our companies. Through four years of groundbreaking research to include data collected from the State of DevOps reports conducted with Puppet, Dr. Nicole Forsgren, Jez Humble, and Gene Kim set out to find a way to measure software delivery performance?and what drives it?using rigorous statistical methods. This book presents both the findings and the science behind that research, making the information accessible for readers to apply in their own organizations. Readers will discover how to measure the performance of their teams, and what capabilities they should invest in to drive higher performance. This book is ideal for management at every level. The overwhelming majority of a software system's lifespan is spent in use, not in design or implementation. So, why does conventional wisdom insist that software engineers focus primarily on the design and development of large-scale computing systems? In this collection of essays and articles, key members of Google's Site Reliability Team explain how and why their commitment to the entire lifecycle has

File Type PDF Creating A Software Engineering Culture

enabled the company to successfully build, deploy, monitor, and maintain some of the largest software systems in the world. You'll learn the principles and practices that enable Google engineers to make systems more scalable, reliable, and efficient—lessons directly applicable to your organization. This book is divided into four sections:

Introduction—Learn what site reliability engineering is and why it differs from conventional IT industry practices
Principles—Examine the patterns, behaviors, and areas of concern that influence the work of a site reliability engineer (SRE)

Practices—Understand the theory and practice of an SRE's day-to-day work: building and operating large distributed computing systems

Management—Explore Google's best practices for training, communication, and meetings that your organization can use

Many claims are made about how certain tools, technologies, and practices improve software development. But which claims are verifiable, and which are

File Type PDF Creating A Software Engineering Culture

merely wishful thinking? In this book, leading thinkers such as Steve McConnell, Barry Boehm, and Barbara Kitchenham offer essays that uncover the truth and unmask myths commonly held among the software development community. Their insights may surprise you. Are some programmers really ten times more productive than others? Does writing tests first help you develop better code faster? Can code metrics predict the number of bugs in a piece of software? Do design patterns actually make better software? What effect does personality have on pair programming? What matters more: how far apart people are geographically, or how far apart they are in the org chart? Contributors include: Jorge Aranda Tom Ball Victor R. Basili Andrew Begel Christian Bird Barry Boehm Marcelo Cataldo Steven Clarke Jason Cohen Robert DeLine Madeline Diep Hakan Erdogmus Michael Godfrey Mark Guzdial Jo E. Hannay Ahmed E. Hassan Israel Herraiz Kim Sebastian Herzig Cory Kapser Barbara Kitchenham Andrew Ko Lucas Layman Steve McConnell Tim Menzies Gail Murphy Nachi Nagappan

File Type PDF Creating A Software Engineering Culture

Thomas J. Ostrand Dewayne Perry Marian
Petre Lutz Prechelt Rahul Premraj
Forrest Shull Beth Simon Diomidis
Spinellis Neil Thomas Walter Tichy
Burak Turhan Elaine J. Weyuker Michele
A. Whitecraft Laurie Williams Wendy M.
Williams Andreas Zeller Thomas
Zimmermann

Software Architect's Handbook

Measure What Matters

Applying the SEMAT Kernel

Organizing Business and Technology

Teams for Fast Flow

Data Pipelines Pocket Reference

The Hidden Strengths of Learning and
Succeeding at Your Own Pace

The Science of Lean Software and

DevOps: Building and Scaling High

Performing Technology Organizations

Over the last fifteen years, every major aspect of our lives has changed because of Facebook. You may not like Facebook, but you can't deny its success. And to a large degree, that success stems from the "move fast" ethos. The entire culture of Facebook is built for speed. Move Fast is an exploration of modern software strategies and tactics through the lens of Facebook. Relying on in-depth interviews with more than two dozen Facebook engineers, this book explores

File Type PDF Creating A Software Engineering Culture

the product strategy, cultural principles, and technologies that made Facebook the dominant social networking company. Most importantly, Move Fast investigates how you can apply those strategies to your creative projects. It's not easy to build a software company, but once you know how to move fast, your company will be prepared to build a strategy that benefits from the world's rapid changes, rather than suffering from them. Software startups make global headlines every day. As technology companies succeed and grow, so do their engineering departments. In your career, you'll may suddenly get the opportunity to lead teams: to become a manager. But this is often uncharted territory. How can you decide whether this career move is right for you? And if you do, what do you need to learn to succeed? Where do you start? How do you know that you're doing it right? What does "it" even mean? And isn't management a dirty word? This book will share the secrets you need to know to manage engineers successfully. Going from engineer to manager doesn't have to be intimidating. Engineers can be managers, and fantastic ones at that. Cast aside the rhetoric and focus on practical, hands-on techniques and tools. You'll become an effective and supportive team leader that your staff will look up to. Start with your transition to being a manager and see how that compares to being an engineer. Learn how to better organize information, feel productive, and delegate,

File Type PDF Creating A Software Engineering Culture

but not micromanage. Discover how to manage your own boss, hire and fire, do performance and salary reviews, and build a great team. You'll also learn the psychology: how to ship while keeping staff happy, coach and mentor, deal with deadline pressure, handle sensitive information, and navigate workplace politics. Consider your whole department. How can you work with other teams to ensure best practice? How do you help form guilds and committees and communicate effectively? How can you create career tracks for individual contributors and managers? How can you support flexible and remote working? How can you improve diversity in the industry through your own actions? This book will show you how. Great managers can make the world a better place. Join us.

Great technology alone is rarely sufficient today to ensure a product's success. At Microsoft, scenario-focused engineering is a customer-centric, iterative approach used to design and deliver the deeper experiences and emotional engagement customers demand in new products. In this book, you'll discover the proven practices and lessons learned from real-world implementations of this approach, including: Why design matters: Understand a competitive landscape where customers are no longer satisfied by products that are merely useful, but respond instead to products they crave using. What it means to be customer focused: Recognize that you are not the customer, understand customers can have

File Type PDF Creating A Software Engineering Culture

difficulty articulating what they want, and apply techniques that uncover their unspoken needs. How to iterate effectively: Implement a development system that is flexible enough to respond to early and continuous feedback, and enables experimentation with multiple ideas and feedback loops simultaneously. How to bridge the culture gap: In an engineering environment traditionally rooted in strong analytics, the ideas and practices for scenario-focused engineering may not be intuitive. Learn how to change team mindset from deciding what a product, service, or device will do, to discovering what customers actually want and what will work for them in real-life scenarios. Connections with Lean and Agile approaches: See the connections, gaps, and overlaps among the Lean, Agile, and Scenario-Focused Engineering methodologies, and achieve a more holistic view of software development.

This book has assembled a guide that will help you hire, motivate, and mentor a software development team that functions at the highest level. Their rules of thumb and coaching advice form a great blueprint for new and experienced software engineering managers alike. All too often, software development is deemed unmanageable. The news is filled with stories of projects that have run catastrophically over schedule and budget.

Ask Your Developer

The Essence of Software Engineering

Foundations of Machine Learning, second edition

Musings on Linux and Open Source by an Accidental Revolutionary

39 Engineering Challenges

Organizational Patterns of Agile Software Development

Late Bloomers

#1 New York Times Bestseller Legendary venture capitalist John Doerr reveals how the goal-setting system of Objectives and Key Results (OKRs) has helped tech giants from Intel to Google achieve explosive growth—and how it can help any organization thrive. In the fall of 1999, John Doerr met with the founders of a start-up whom he'd just given \$12.5 million, the biggest investment of his career. Larry Page and Sergey Brin had amazing technology, entrepreneurial energy, and sky-high ambitions, but no real business plan. For Google to change the world (or even to survive), Page and Brin had to learn how to make tough choices on priorities while keeping their team on track. They'd have to know when to pull the plug on losing propositions, to fail fast. And they needed timely, relevant data to track their progress—to measure what mattered. Doerr taught them about

a proven approach to operating excellence: Objectives and Key Results. He had first discovered OKRs in the 1970s as an engineer at Intel, where the legendary Andy Grove ("the greatest manager of his or any era") drove the best-run company Doerr had ever seen. Later, as a venture capitalist, Doerr shared Grove's brainchild with more than fifty companies. Wherever the process was faithfully practiced, it worked. In this goal-setting system, objectives define what we seek to achieve; key results are how those top-priority goals will be attained with specific, measurable actions within a set time frame. Everyone's goals, from entry level to CEO, are transparent to the entire organization. The benefits are profound. OKRs surface an organization's most important work. They focus effort and foster coordination. They keep employees on track. They link objectives across silos to unify and strengthen the entire company. Along the way, OKRs enhance workplace satisfaction and boost retention. In Measure What Matters, Doerr shares a broad range of first-person, behind-the-scenes case studies,

with narrators including Bono and Bill Gates, to demonstrate the focus, agility, and explosive growth that OKRs have spurred at so many great organizations. This book will help a new generation of leaders capture the same magic.

In Team Topologies DevOps consultants Matthew Skelton and Manuel Pais share secrets of successful team patterns and interactions to help readers choose and evolve the right team patterns for their organization, making sure to keep the software healthy and optimize value streams. Team Topologies will help readers discover:

- Team patterns used by successful organizations.**
- Common team patterns to avoid with modern software systems.**
- When and why to use different team patterns**
- How to evolve teams effectively.**
- How to split software and align to teams.**

SEMAT (Software Engineering Methods and Theory) is an international initiative designed to identify a common ground, or universal standard, for software engineering. It is supported by some of the most distinguished contributors to the field. Creating a simple language to describe methods and practices, the

SEMAT team expresses this common ground as a kernel-or framework-of elements essential to all software development. The Essence of Software Engineering introduces this kernel and shows how to apply it when developing software and improving a team's way of working. It is a book for software professionals, not methodologists. Its usefulness to development team members, who need to evaluate and choose the best practices for their work, goes well beyond the description or application of any single method.

“Software is both a craft and a science, both a work of passion and a work of principle. Writing good software requires both wild flights of imagination and creativity, as well as the hard reality of engineering tradeoffs. This book is an attempt at describing that balance.”

—Robert Martin (unclebob) “The work of Ivar Jacobson and his colleagues, started as part of the SEMAT initiative, has taken a systematic approach to identifying a ‘kernel’ of software engineering principles and practices that have stood the test of time and recognition.” —Bertrand Meyer “The

software development industry needs and demands a core kernel and language for defining software development practices—practices that can be mixed and matched, brought on board from other organizations; practices that can be measured; practices that can be integrated; and practices that can be compared and contrasted for speed, quality, and price. This thoughtful book gives a good grounding in ways to think about the problem, and a language to address the need, and every software engineer should read it.” —Richard Soley

I am a Software Engineer and I am in Charge is a real-world, practical book that helps you increase your impact and satisfaction at work no matter who you work with. Each of the 7 chapters has the following structure specifically designed to generate insight and move you to action. Why it matters A brief introduction to the chapter that offers questions for you to experiment with your current belief about the topic of the chapter. For example, if you believe you can't ask a colleague you admire to be your mentor, then what could you do if you changed that belief? The story A fictional story

following the protagonist, Sandrine who left her company to get a higher-level role and found that despite the "promotion" everything still feels the same, the people around her are clueless. In each chapter, Sandrine learns something from the people she interacts with that gets her thinking in a new way enabling her to take different actions. Sandrine is not perfect though, she makes slip-ups, promises to change but goes back to old habits, plans for things a certain way only to discover it doesn't play out that way-just like in real life. What do we learn from the story Here we talk about the lesson from the story, and ask you, the reader, what you will do with your new knowledge and insights. The experiments At the end of each chapter, there are 3 experiments for you to try. You can choose to do one or more of them to see what happens when you put yourself in Sandrine's shoes. Follow Sandrine on her journey to see for yourself how she solved her problems and increased her impact and satisfaction and in the process find a way to increase yours. By the end of the book you'll have learned: How your words

influence your actions
How to prosper from feedback
How to set goals that inspire
How to work with others to create a better solution
How to use failure as a data point to inform your learnin

What Really Works, and Why We Believe It

How Google, Bono, and the Gates Foundation Rock the World with OKRs
Research Software Engineering with Python

Building Mobile Apps at Scale
Scenario-Focused Engineering
Managing the Unmanageable

The Book that Helps Increase Your Impact and Satisfaction at Work

Jeff Lawson, developer turned CEO of Twilio (one of Bloomberg Businessweek's Top 50 Companies to Watch 2021), creates a new playbook for unleashing the full potential of software developers in any organization, showing how to help management utilize this coveted valuable workforce to enable growth, solve a wide range of business problems, and drive digital transformation. From banking and retail to insurance and finance, every industry is turning digital, and every company needs the best software to win the hearts and minds of customers. The landscape has shifted from the classic build vs. buy question, to one of build vs. die. Companies have to get this right to survive. But how do they make this transi

File Type PDF Creating A Software Engineering Culture

Software developers are sought after, highly paid, and desperately needed to compete in the modern, digital economy. Yet most companies treat them like digital factory workers without really understanding how to unleash their full potential. Lawson argues that developers are the creative workforce who can solve major business problems and create hit products for customers—not just grind through rote tasks. From Google and Amazon, to one-person online software companies—companies that bring software developers as partners are winning. Lawson shows how leaders who build industry changing software products consistently do three things well. First, they understand why software developers matter more than ever. Second, they understand developers and know how to motivate them. And third, they invest in their developers' success. As a software developer and public company CEO, Lawson uses his unique position to bridge the language and to help executives use with the unique culture of high performing creative software developers. *Ask Your Developer* is a toolkit to help business leaders, product managers, technical leaders, software developers, and executives achieve their common goal—building great digital products and experiences. How to compete in the digital economy? In short: *Ask Your Developer*.

From the creator of the popular website *Ask a Manager* and New York's work-advice columnist comes a witty, practical guide to 200 difficult professional conversations—featuring all-new advice! There's a reason

File Type PDF Creating A Software Engineering Culture

Alison Green has been called “the Dear Abby of the work world.” Ten years as a workplace-advice columnist have taught her that people avoid awkward conversations in the office because they simply don’t know what to say. Thankfully, Green does—and in this incredibly helpful book, she tackles the tough discussions you may need to have during your career. You’ll learn what to say when coworkers push their work on you—then take credit for you accidentally trash-talk someone in an email then hit “reply all” • you’re being micromanaged—or not being managed at all • you catch a colleague in a lie • your boss seems unhappy with your work • your cubemate’s loud speakerphone is making you homicidal • you got drunk at the holiday party

Praise for *Ask a Manager* “A must-read for anyone who works . . . [Alison Green’s] advice boils down to the idea that you should be professional (even when others are not) and that communicating in a straightforward manner with candor and kindness will get you far, no matter where you work.”—Booklist (starred review) “The author’s friendly, warm, no-nonsense writing is a pleasure to read, and her advice can be widely applied to relationships in all areas of readers’ lives. Ideal for anyone new to the job market or new to management, or anyone hoping to improve their work experience.”—Library Journal (starred review) “I am a huge fan of Alison Green’s *Ask a Manager* column. This book is even better. It teaches us how to deal with many of the most vexing big and little problems in our workplaces—and to do so with grace, confidence, and

File Type PDF Creating A Software Engineering Culture

sense of humor.”—Robert Sutton, Stanford professor and author of *The No Asshole Rule* and *The Asshole Survival Guide* “Ask a Manager is the ultimate playbook for navigating the traditional workforce in a diplomatic but firm way.”—Erin Lowry, author of *Broke Millennial: Stop Scraping By and Get Your Financial Life Together*

A comprehensive guide to exploring software architecture concepts and implementing best practices Key Features Enhance your skills to grow your career as a software architect Design efficient software architectures using patterns and best practices Learn how software architecture relates to an organization as well as software development methodology Book Description *The Software Architect’s Handbook* is a comprehensive guide to help developers, architects, and senior programmers advance their career in the software architecture domain. This book takes you through all the important concepts, right from design principles to different considerations at various stages of your career in software architecture. The book begins by covering the fundamentals, benefits, and purpose of software architecture. You will discover how software architecture relates to an organization, followed by identifying its significant quality attributes. Once you have covered the basics, you will explore design patterns, best practices, and paradigms for efficient software development. The book discusses which factors you need to consider for performance and security enhancements. You will learn to write documentation for your architectures and make appropriate decisions when

File Type PDF Creating A Software Engineering Culture

considering DevOps. In addition to this, you will explore how to design legacy applications before understanding how to create software architectures that evolve as the market, business requirements, frameworks, tools, and best practices change over time. By the end of this book you will not only have studied software architecture concepts but also built the soft skills necessary to grow in this field. What you will learn

- Design software architectures using patterns and best practices
- Explore the different considerations for designing software architecture
- Discover what it takes to continuously improve as a software architect
- Create loosely coupled systems that can support change
- Understand DevOps and how it affects software architecture
- Integrate, refactor, and re-architect legacy applications

Who this book is for
The Software Architect's Handbook is for you if you are a software architect, chief technical officer (CTO), or senior developer looking to gain a firm grasp of software architecture.

Open source provides the competitive advantage in the Internet Age. According to the August Forrester Report, 56 percent of IT managers interviewed at Global 2,500 companies are already using some type of open source software in their infrastructure and another 6 percent will install it in the next two years. This revolutionary model for collaborative software development is being embraced and studied by many of the biggest players in the high-tech industry, from Sun Microsystems to IBM to Intel.

Cathedral & the Bazaar is a must for anyone who cares

File Type PDF Creating A Software Engineering Culture

about the future of the computer industry or the dynamism of the information economy. Already, billions of dollars have been made and lost based on the ideas in this book. Its conclusions will be studied, debated, and implemented for years to come. According to Bob Young, "This is Eric Raymond's great contribution to the success of the open source revolution, to the adoption of Linux-based operating systems, and to the success of open source and the companies that supply them." The interest in open source software development has grown enormously in the past year. This revised and expanded paperback edition includes new material on open source developments in 1999 and 2000. Raymond's clear and effective writing style accurately describing the benefits of open source software has been key to its success. With major vendors creating acceptance for open source within their companies, independent vendors will become the open source story in 2001.

The Lean Mindset

The Effective Engineer

Engineering Culture

Become an Effective Software Engineering Manager

Software Development Pearls

Better Productivity Through Collaboration

How to Be the Leader Your Development Team Needs

"This remarkable book combines practical advice, ready-to-use techniques, and a deep understanding of why this is the right way to develop software. I have seen software

teams transformed by the ideas in this book." --Mike Cohn, author of Agile Estimating and Planning "As a lean practitioner myself, I have loved and used their first book for years. When this second book came out, I was delighted that it was even better. If you are interested in how lean principles can be useful for software development organizations, this is the book you are looking for. The Poppendiecks offer a beautiful blend of history, theory, and practice." --Alan Shalloway, coauthor of Design Patterns Explained "I've enjoyed reading the book very much. I feel it might even be better than the first lean book by Tom and Mary, while that one was already exceptionally good! Mary especially has a lot of knowledge related to lean techniques in product development and manufacturing. It's rare that these techniques are actually translated to software. This is something no other book does well (except their first book)." --Bas Vodde "The new book by Mary and Tom Poppendieck provides a well-written and comprehensive introduction to lean principles and selected practices for software managers and engineers. It illustrates the application of the values and practices with well-suited success

stories. I enjoyed reading it." --Roman Pichler "In Implementing Lean Software Development, the Poppendiecks explore more deeply the themes they introduced in Lean Software Development. They begin with a compelling history of lean thinking, then move to key areas such as value, waste, and people. Each chapter includes exercises to help you apply key points. If you want a better understanding of how lean ideas can work with software, this book is for you." --Bill Wake, independent consultant

In 2003, Mary and Tom Poppendieck's Lean Software Development introduced breakthrough development techniques that leverage Lean principles to deliver unprecedented agility and value. Now their widely anticipated sequel and companion guide shows exactly how to implement Lean software development, hands-on. This new book draws on the Poppendiecks' unparalleled experience helping development organizations optimize the entire software value stream. You'll discover the right questions to ask, the key issues to focus on, and techniques proven to work. The authors present case studies from leading-edge software organizations, and offer practical exercises for jumpstarting your own Lean initiatives. Managing to extend, nourish,

and leverage agile practices Building true development teams, not just groups Driving quality through rapid feedback and detailed discipline Making decisions Just-in-Time, but no later Delivering fast: How PatientKeeper delivers 45 rock-solid releases per year Making tradeoffs that really satisfy customers Implementing Lean Software Development is indispensable to anyone who wants more effective development processes--managers, project leaders, senior developers, and architects in enterprise IT and software companies alike.

For courses in Advanced Software Engineering or Object-Oriented Design. This book covers the human and organizational dimension of the software improvement process and software project management - whether based on the CMM or ISO 9000 or the Rational Unified Process. Drawn from a decade of research, it emphasizes common-sense practices. Its principles are general but concrete; every pattern is its own built-in example. Historical supporting material from other disciplines is provided. Though even pattern experts will appreciate the depth and currency of the material, it is self-contained and well-suited for the layperson.

In Improving Software Development Productivity, legendary software engineering expert Dr. Randall Jensen introduces a proven quantitative approach to achieving high productivity through management support, the ability to communicate, and technology. Jensen demonstrates how to measure organizational capacity and productivity, and use that information to build more accurate estimates and schedules -- and, more broadly, to improve many facets of developer and team performance. Students will learn to quantitatively predict the productivity impact of management decisions related to personnel and management style, development environment, product constraints, technology, development systems, and more.

Data pipelines are the foundation for success in data analytics. Moving data from numerous diverse sources and transforming it to provide context is the difference between having data and actually gaining value from it. This pocket reference defines data pipelines and explains how they work in today's modern data stack. You'll learn common considerations and key decision points when implementing pipelines, such as batch versus streaming data ingestion and build

versus buy. This book addresses the most common decisions made by data professionals and discusses foundational concepts that apply to open source frameworks, commercial products, and homegrown solutions. You'll learn: What a data pipeline is and how it works How data is moved and processed on modern data infrastructure, including cloud platforms Common tools and products used by data engineers to build pipelines How pipelines support analytics and reporting needs Considerations for pipeline maintenance, testing, and alerting

Team Topologies

Move Fast

How Google Tests Software

A Software Developer's Guide to Working Well with Others

How Google Runs Production Systems

How to Harness the Power of Software

Developers and Win in the 21st Century

Making Software

Software development consultant Wieggers describes various formal and informal methods for conducting a peer review program, such as pair programming, team reviews, the "walkthrough," and the ad hoc review. The main part of the text is devoted to the various stages of the technique of inspection. Coverage extends to the social issues involved in critiquing the work of others and overcoming resistance to reviews. c. Book News Inc.

This is the digital version of the printed book (Copyright ©

File Type PDF Creating A Software Engineering Culture

1996). Written in a remarkably clear style, *Creating a Software Engineering Culture* presents a comprehensive approach to improving the quality and effectiveness of the software development process. In twenty chapters spread over six parts, Wiegers promotes the tactical changes required to support process improvement and high-quality software development. Throughout the text, Wiegers identifies scores of culture builders and culture killers, and he offers a wealth of references to resources for the software engineer, including seminars, conferences, publications, videos, and on-line information. With case studies on process improvement and software metrics programs and an entire part on action planning (called "What to Do on Monday"), this practical book guides the reader in applying the concepts to real life. Topics include software culture concepts, team behaviors, the five dimensions of a software project, recognizing achievements, optimizing customer involvement, the project champion model, tools for sharing the vision, requirements traceability matrices, the capability maturity model, action planning, testing, inspections, metrics-based project estimation, the cost of quality, and much more!

Principles from Part 1 Never let your boss or your customer talk you into doing a bad job. People need to feel the work they do is appreciated. Ongoing education is every team member's responsibility. Customer involvement is the most critical factor in software quality. Your greatest challenge is sharing the vision of the final product with the customer. Continual improvement of your software development process is both possible and essential. Written software development procedures can help build a shared culture of best practices. Quality is the top priority; long-term productivity is a natural consequence of high quality. Strive to have a peer, rather than a customer, find a defect. A key

File Type PDF Creating A Software Engineering Culture

to software quality is to iterate many times on all development steps except coding: Do this once. Managing bug reports and change requests is essential to controlling quality and maintenance. If you measure what you do, you can learn to do it better. You can't change everything at once. Identify those changes that will yield the greatest benefits, and begin to implement them next Monday. Do what makes sense; don't resort to dogma.

Companies from startups to corporate giants face massive amounts of disruption today. Now more than ever, organizations need nimble and responsive leaders who know how to exploit the opportunities that change brings. In this insightful book, Jean Dahl, a senior executive and expert in the Lean mindset and its methods, demonstrates why you need to embrace Modern Lean principles and thinking to redefine leadership in this age of digital disruption in order to continuously evolve the Lean enterprise. Drawing on nearly three decades of corporate and consulting experience, Ms. Dahl lays out a new holistic framework for developing Modern Lean leaders. Through personal experiences and compelling real-world case studies, she explains specific steps necessary for you and your company to proactively understand and respond to change. Understand the leadership challenges Lean leaders face in our 21st century global economy Explore the six dimensions of the Modern Lean Framework™ Learn and apply the nine steps necessary to become a Lean leader Use Modern Lean methods to build a culture of continuous learning that can be sustained and maintained within your organization Seize competitive advantage by embracing Modern Lean to build an enterprise that understands how to respond to disruption

Writing for students at all levels of experience, Farley illuminates durable principles at the heart of effective

File Type PDF Creating A Software Engineering Culture

software development. He distills the discipline into two core exercises: first, learning and exploration, and second, managing complexity. For each, he defines principles that can help students improve everything from their mindset to the quality of their code, and describes approaches proven to promote success. Farley's ideas and techniques cohere into a unified, scientific, and foundational approach to solving practical software development problems within realistic economic constraints. This general, durable, and pervasive approach to software engineering can help students solve problems they haven't encountered yet, using today's technologies and tomorrow's. It offers students deeper insight into what they do every day, helping them create better software, faster, with more pleasure and personal fulfillment.

*Control and Commitment in a High-tech Corporation
Ensuring Success and Developing a Framework for Leadership*

Become a successful software architect by implementing effective architecture concepts

I Am a Software Engineer and I Am in Charge

Peer Reviews in Software

How to Leverage Your Efforts in Software Engineering to Make a Disproportionate and Meaningful Impact

Effective Leadership and Quantitative Methods in Software Management

While there is a lot of appreciation for backend and distributed systems challenges, there tends to be less empathy for why mobile development is hard when done at scale. This book collects challenges engineers face when

File Type PDF Creating A Software Engineering Culture

building iOS and Android apps at scale, and common ways to tackle these. By scale, we mean having numbers of users in the millions and being built by large engineering teams. For mobile engineers, this book is a blueprint for modern app engineering approaches. For non-mobile engineers and managers, it is a resource with which to build empathy and appreciation for the complexity of world-class mobile engineering. The book covers iOS and Android mobile app challenges on these dimensions: Challenges due to the unique nature of mobile applications compared to the web, and to the backend. App complexity challenges. How do you deal with increasingly complicated navigation patterns? What about non-deterministic event combinations? How do you localize across several languages, and how do you scale your automated and manual tests? Challenges due to large engineering teams. The larger the mobile team, the more challenging it becomes to ensure a consistent architecture. If your company builds multiple apps, how do you balance not

File Type PDF Creating A Software Engineering Culture

rewriting everything from scratch while moving at a fast pace, over waiting on "centralized" teams? Cross-platform approaches. The tooling to build mobile apps keeps changing. New languages, frameworks, and approaches that all promise to address the pain points of mobile engineering keep appearing. But which approach should you choose?

Flutter, React Native, Cordova? Native apps? Reuse business logic written in Kotlin, C#, C++ or other languages? What engineering approaches do "world-class" mobile engineering teams choose in non-functional aspects like code quality, compliance, privacy, compliance, or with experimentation, performance, or app size?

This is the digital version of the printed book (Copyright (c) 1996). Written in a remarkably clear style, "Creating a Software Engineering Culture" presents a comprehensive approach to improving the quality and effectiveness of the software development process. In twenty chapters spread over six parts, Wieggers promotes the tactical changes required to support process improvement and high-quality software development.

File Type PDF Creating A Software Engineering Culture

Throughout the text, Wiegers identifies scores of culture builders and culture killers, and he offers a wealth of references to resources for the software engineer, including seminars, conferences, publications, videos, and on-line information. With case studies on process improvement and software metrics programs and an entire part on action planning (called "What to Do on Monday"), this practical book guides the reader in applying the concepts to real life. Topics include software culture concepts, team behaviors, the five dimensions of a software project, recognizing achievements, optimizing customer involvement, the project champion model, tools for sharing the vision, requirements traceability matrices, the capability maturity model, action planning, testing, inspections, metrics-based project estimation, the cost of quality, and much more Principles from Part 1 Never let your boss or your customer talk you into doing a bad job. People need to feel the work they do is appreciated. Ongoing education is every team member's responsibility. Customer

File Type PDF Creating A Software Engineering Culture

involvement is the most critical factor in software quality. Your greatest challenge is sharing the vision of the final product with the customer.

Continual improvement of your software development process is both possible and essential. Written software development procedures can help build a shared culture of best practices.

Quality is the top priority; long-term productivity is a natural consequence of high quality. Strive to have a peer, rather than a customer, find a defect.

A key to software quality is to iterate many times on all development steps except coding: Do this once. Managing bug reports and change requests is essential to controlling quality and maintenance. If you measure what you do, you can learn to do it better. You can't change everything at once.

Identify those changes that will yield the greatest benefits, and begin to implement them next Monday. Do what makes sense; don't resort to dogma.

Drawing on 20+ years helping software teams succeed in nearly 150 organizations, Karl Wieggers presents 60 concise lessons and practical

File Type PDF Creating A Software Engineering Culture

recommendations students can apply to all kinds of projects, regardless of application domain, technology, development lifecycle, or platform infrastructure. Embodying both wisdom for deeper understanding and guidance for practical use, this book represents an invaluable complement to the technical nuts and bolts software developers usually study. *Software Development Pearls* covers multiple crucial domains of project success: requirements, design, project management, culture and teamwork, quality, and process improvement. Each chapter suggests several first steps and next steps to help you begin immediately applying the author's hard-won lessons--and writing code that is more successful in every way that matters.

A groundbreaking exploration of what it means to be a late bloomer in a culture obsessed with SAT scores and early success, and how finding one's way later in life can be an advantage to long-term achievement and happiness. We live in a society where kids and parents are obsessed with early

File Type PDF Creating A Software Engineering Culture

achievement, from getting perfect scores on SATs to getting into Ivy League colleges to landing an amazing job at Google or Facebook--or even better, creating a startup with the potential to be the next Google or Facebook or Uber. We see software coders becoming millionaires or even billionaires before age 30 and feel we are failing if we are not one of them. But there is good news. A lot of us--most of us--do not explode out of the gates in life. That was true for author Rich Karlgaard, who had a mediocre academic career at Stanford (which he got into by a fluke), and after graduating, worked as a dishwasher, nightwatchman, and typing temp before finally finding the inner motivation and drive that ultimately led him to start up a high-tech magazine in Silicon Valley, and eventually to become the publisher of Forbes magazine. There is a scientific explanation for why so many of us bloom later in life. The executive function of our brains doesn't mature until age 25--and later for some. In fact our brain's capabilities peak at different

File Type PDF Creating A Software Engineering Culture

ages. We actually enjoy multiple periods of blooming in our lives. Based on several years of research, personal experience, and interviews with neuroscientists and psychologists, and countless people at different stages of their careers, Bloom reveals how and when we achieve full potential--and why an algorithmic acuity in math is such an anomaly in terms of career success.

Agile Leadership Toolkit

Debugging Teams

Software Engineering at Google

Site Reliability Engineering

Creating Software Eng Cul

The Cathedral & the Bazaar

Lessons from Fifty Years of Software Experience

"Engineering Culture" is an award-winning ethnography of the engineering division of a large American high-tech corporation. Now, this influential book - which has been translated into Japanese, Italian and Hebrew - has been revised to bring it up to date. In "Engineering Culture", Gideon Kunda offers a critical analysis of an American company's well-known and widely emulated "corporate culture." Kunda uses detailed descriptions of everyday interactions and rituals in which the culture is brought to life, excerpts from

in-depth interviews and a wide variety of corporate texts to vividly portray managerial attempts to design and impose the culture and the ways in which it is experienced by members of the organization. The company's management, Kunda reveals, uses a variety of methods to promulgate what it claims is a non-authoritarian, informal, and flexible work environment that enhances and rewards individual commitment, initiative, and creativity while promoting personal growth. The author demonstrates, however, that these pervasive efforts mask an elaborate and subtle form of normative control in which the members' minds and hearts become the target of corporate influence. Kunda carefully dissects the impact this form of control has on employees' work behavior and on their sense of self. In the conclusion written especially for this edition, Kunda reviews the company's fortunes in the years that followed publication of the first edition, reevaluates the arguments in the book, and explores the relevance of corporate culture and its management today

In a perfect world, software engineers who produce the best code are the most successful. But in our perfectly messy world, success also depends on how you work with people to get your job done. In this highly

entertaining book, Brian Fitzpatrick and Ben Collins-Sussman cover basic patterns and anti-patterns for working with other people, teams, and users while trying to develop software. This is valuable information from two respected software engineers whose popular series of talks—including "Working with Poisonous People"—has attracted hundreds of thousands of followers. Writing software is a team sport, and human factors have as much influence on the outcome as technical factors. Even if you've spent decades learning the technical side of programming, this book teaches you about the often-overlooked human component. By learning to collaborate and investing in the "soft skills" of software engineering, you can have a much greater impact for the same amount of effort. Team Geek was named as a Finalist in the 2013 Jolt Awards from Dr. Dobb's Journal. The publication's panel of judges chose five notable books, published during a 12-month period ending June 30, that every serious programmer should read.

Writing and running software is now as much a part of science as telescopes and test tubes, but most researchers are never taught how to do either well. As a result, it takes them longer to accomplish simple tasks than it should, and it is harder for

them to share their work with others than it needs to be. This book introduces the concepts, tools, and skills that researchers need to get more done in less time and with less pain. Based on the practical experiences of its authors, who collectively have spent several decades teaching software skills to scientists, it covers everything graduate-level researchers need to automate their workflows, collaborate with colleagues, ensure that their results are trustworthy, and publish what they have built so that others can build on it. The book assumes only a basic knowledge of Python as a starting point, and shows readers how it, the Unix shell, Git, Make, and related tools can give them more time to focus on the research they actually want to do. Research Software Engineering with Python can be used as the main text in a one-semester course or for self-guided study. A running example shows how to organize a small research project step by step; over a hundred exercises give readers a chance to practice these skills themselves, while a glossary defining over two hundred terms will help readers find their way through the terminology. All of the material can be re-used under a Creative Commons license, and all royalties from sales of the book will be donated to The Carpentries, an organization

that teaches foundational coding and data science skills to researchers worldwide. 2012 Jolt Award finalist! Pioneering the Future of Software Test Do you need to get it right, too? Then, learn from Google. Legendary testing expert James Whittaker, until recently a Google testing leader, and two top Google experts reveal exactly how Google tests software, offering brand-new best practices you can use even if you're not quite Google's size...yet! Breakthrough Techniques You Can Actually Use Discover 100% practical, amazingly scalable techniques for analyzing risk and planning tests...thinking like real users...implementing exploratory, black box, white box, and acceptance testing...getting usable feedback...tracking issues...choosing and creating tools...testing "Docs & Mocks," interfaces, classes, modules, libraries, binaries, services, and infrastructure...reviewing code and refactoring...using test hooks, presubmit scripts, queues, continuous builds, and more. With these techniques, you can transform testing from a bottleneck into an accelerator-and make your whole organization more productive! Ask the Right Questions Ask a Manager Building software that makes research

possible

How Facebook Builds Software

Improving Software Development

Productivity

Modern Software Engineering

A Practical Guide

Develop faster with DevOps DevOps embraces a culture of unifying the creation and distribution of technology in a way that allows for faster release cycles and more resource-efficient product updating. DevOps For Dummies provides a guidebook for those on the development or operations side in need of a primer on this way of working. Inside, DevOps evangelist Emily Freeman provides a roadmap for adopting the management and technology tools, as well as the culture changes, needed to dive head-first into DevOps. Identify your organization's needs Create a DevOps framework Change your organizational structure Manage projects in the DevOps world DevOps For Dummies is essential reading for developers and operations professionals in the early stages of DevOps adoption. Practical, Proven Tools for Leading and Empowering High-Performing Agile Teams

File Type PDF Creating A Software Engineering Culture

A leader is like a farmer, who doesn't grow crops by pulling them but instead creates the perfect environment for the crops to grow and thrive. If you lead in organizations that have adopted agile methods, you know it's crucial to create the right environment for your agile teams. Traditional tools such as Gantt charts, detailed plans, and internal KPIs aren't adequate for complex and fast-changing markets, but merely trusting employees and teams to self-manage is insufficient as well. In *Agile Leadership Toolkit*, longtime agile leader Peter Koning provides a practical and invaluable steering wheel for agile leaders and their teams. Drawing on his extensive experience helping leaders drive more value from agile, Koning offers a comprehensive toolkit for continuously improving your environment, including structures, metrics, meeting techniques, and governance for creating thriving teams that build disruptive products and services. Koning thoughtfully explains how to lead agile teams at large scale and how team members fit into both the team and the wider organization.

File Type PDF Creating A Software Engineering Culture

Architect environments that help teams learn, grow, and flourish for the long term Get timely feedback everyone can use to improve Co-create goals focused on the customer, not the internal organization Help teams brainstorm and visualize the value of their work to the customer Facilitate team ownership and accelerate team learning Support culture change, and design healthier team habits Make bigger changes faster This actionable guide is for leaders at all levels—whether you're supervising your first agile team, responsible for multiple teams, or lead the entire company. Register your book for convenient access to downloads, updates, and/or corrections as they become available. See inside book for details.

Rules, Tools, and Insights for Managing Software People and Teams

Doing What Works to Build Better Software Faster

Creating a Software Engineering Culture Paperback

DevOps For Dummies

Leading Lean

From Concept to Cash