Software Design Document Sample

Document the architecture of your software easily with this highly practical, open-source template. Key FeaturesGet to grips with leveraging the features of arc42 to create insightful documentsLearn the concepts of software architecture documentation through real-world examplesDiscover techniques to create compact, helpful, and easy-to-read documentationBook Description When developers document the architecture of their systems, they often invent their own specific ways of articulating structures, designs, concepts, and decisions. What they need is a template that enables simple and efficient software architecture documentation. arc42 by Example shows how it's done through several real-world examples. Each example in the book, whether it is a chess engine, a huge CRM system, or a cool web system, starts with a brief description of the problem domain and the quality requirements. Then, you'll discover the system context with all the external interfaces. You'll dive into an overview of the solution strategy to implement the building blocks and runtime scenarios. The later chapters also explain various cross-cutting concerns and how they affect other aspects of a program. What you will learnUtilize arc42 to document a system's physical infrastructureLearn how to identify a system's scope and boundariesBreak a system down into building blocks and illustrate the relationships between themDiscover how to describe the runtime behavior of a systemKnow how to document design decisions and their reasonsExplore the risks and technical debt of your systemWho this book is for This book is for software developers and solutions architects who are looking for an easy, open-source tool to document their systems. It is a useful reference for those who are already using arc42. If you are new to arc42, this book is a great learning resource. For those of you who want to write better technical documentation will benefit from the general concepts covered in this book.

This book contains a refereed collection of thoroughly revised full papers based on the contributions accepted for presentation at the International Workshop on Studies of Software Design, held in conjunction with the 1993 International Conference on Software Engineering, ICSE'93, in Baltimore, Maryland, in May 1993. The emphasis of the 13 papers included is on methods for studying, analyzing, and comparing designs and design methods; the topical focus is primarily on the software architecture level of design and on techniques suitable for dealing with large software systems. The book is organized in sections on architectures, tools, and design methods and opens with a detailed introduction by the volume editor.

Explore the latest Java-based software development techniques and methodologies through the project-based approach in this practical guide. Unlike books that use abstract examples and lots of theory, Real-World Software Development shows you how to develop several relevant projects while learning best practices along the way. With this engaging approach, junior developers capable of writing basic Java code will learn about state-of-the-art software development practices for building modern, robust and maintainable Java software. You'll work with many different software development topics that are often excluded from software develop how-to references. Featuring real-world examples, this book teaches you techniques and methodologies for functional programming, automated testing, security, architecture, and distributed systems.

Author Joseph Dyro has been awarded the Association for the Advancement of Medical Instrumentation (AAMI) Clinical/Biomedical Engineering Achievement Award which recognizes individual excellence and achievement in the clinical engineering and biomedical engineering fields. He has also been awarded the American College of Clinical Engineering 2005 Tom O'Dea Advocacy Award. As the biomedical engineering field expands throughout the world, clinical engineers play an evermore important role as the translator between the worlds of the medical, engineering, and business professionals. They influence procedure and policy at research facilities, universities and private and government agencies including the Food and Drug Administration and the World Health Organization. Clinical Engineers were key players in calming the hysteria over electrical safety in the 1970's and Y2K at the turn of the century and continue to work for medical safety. This title brings together all the important aspects of Clinical Engineering. It provides the reader with prospects for the future of clinical engineering as well as guidelines and standards for best practice around the world. * Clinical Engineers are the safety and quality faciltators in all medical facilities.

Engineering Software

Essentials of Software Engineering

Safety and Reliability Issues

Standardized development of computer software

Lessons Learned from Programming Over Time

8th European Conference, ECSA 2014, Vienna, Austria, August 25-29, 2014, Proceedings

What is this book about? Open source technology enables you to build customized enterprise portal frameworks with more flexibility and fewer limitations. This book explains the fundamentals of a powerful set of open source tools and shows you how to use them. An outstanding team of authors provides a complete tutorial and reference guide to Java Portlet API, Lucene, James, and Slide, taking you step-by-step through constructing and deploying portal applications. You trace the anatomy of a search engine and understand the Lucene query syntax, set up Apache James configuration for a variety of servers, explore object to relational mapping concepts with Jakarta OJB, and acquire many other skills necessary to create J2EE portals uniquely suited to the needs of your organization. Loaded with code-intensive examples of portal applications, this book offers you the know-how to free your development process from the restrictions of pre-packaged solutions. What does this book cover? Here's what you will learn in this book: How to evaluate business requirements and plan the portal How to develop an effective browser environment How to provide a search engine, messaging, database inquiry, and content management services in an integrated portal application How to develop Web services for the portal How to monitor, test, and administer the portal How to create portlet applications compliant with the Java Portlet API How to reduce the possibility of errors while managing the portal to accommodate change How to plan for the next generation application portal Who is this book for? This book is for professional Java developers who have some experience in portal development and want to take advantage of the options offered by open source tools.

The fastest way to get certified for the exams CX-310-252A and CX-310-027. This volume contains tips, tricks, and hints on all the content included in these tests.

This guide will help readers learn how to employ the significant power of use cases to their software development efforts. It provides a practical methodology, presenting key use case concepts.

Praise for the first edition: “This excellent text will be useful to everysystem engineer (SE) regardless of the domain. It covers ALLrelevant SE material and does so in a very clear, methodicalfashion. The breadth and depth of the author's presentation ofSE principles and practices is outstanding.” –Philip Allen This textbook presents a comprehensive, step-by-step guide toSystem Engineering analysis, design, and development via anintegrated set of concepts, principles, practices, andmethodologies. The methods presented in this text apply to any typeof human system -- small, medium, and large organizational systemsand system development projects delivering engineered systems orservices across multiple business sectors such as medical,transportation, financial, educational, governmental, aerospace anddefense, utilities, political, and charity, among others. Provides a common focal point for “bridgingthe gap” between and unifying System Users, System Acquirers,multi-discipline System Engineering, and Project, Functional, andExecutive Management education, knowledge, and decision-making fordeveloping systems, products, or services Each chapter provides definitions of key terms,guiding principles, examples, author’s notes, real-worldexamples, and exercises, which highlight and reinforce key SE&Dconcepts and practices Addresses concepts employed in Model-BasedSystems Engineering (MBSE), Model-Driven Design (MDD), UnifiedModeling Language (UMLTM) / Systems Modeling Language(SysMLTM), and Agile/Spiral/V-Model Development such asuser needs, stories, and use cases analysis; specificationdevelopment; system architecture development; User-Centric SystemDesign (UCSD); interface definition & control; systemintegration & test; and Verification & Validation(V&V) Highlights/introduces a new 21st Century SystemsEngineering & Development (SE&D) paradigm that is easy tounderstand and implement. Provides practices that are critical stagingpoints for technical decision making such as Technical StrategyDevelopment; Life Cycle requirements; Phases, Modes, & States;SE Process; Requirements Derivation; System ArchitectureDevelopment, User-Centric System Design (UCSD); EngineeringStandards, Coordinate Systems, and Conventions; et al. Thoroughly illustrated, with end-of-chapter exercises andnumerous case studies and examples, Systems EngineeringAnalysis, Design, and Development, Second Edition is a primarytextbook for multi-discipline, engineering, system analysis, andproject management undergraduate/graduate level students and avaluable reference for professionals.

SOFTWARE DESIGN, ARCHITECTURE AND ENGINEERING

Tackling Complexity in the Heart of Software

Building Reliable Systems

Real-World Software Development

Concepts, Principles, and Practices

Views and Beyond

A Project-Driven Guide to Fundamentals in Java

"Ali Arya guides you in a fantastic journey full of creativity in a coherent way that allows the traveler to learn and build up over the knowledge acquired in previous chapters until the reader accomplishes skills to develop solutions using programming." — Andrés A. Navarro Newball, Professor, Pontificia Universidad Javeriana, Cali, Colombia "An excellent book that teaches programming and software development the way it should be done: independent from a specific implementation language and focusing on the main principles that are fundamental and substantive to any kind of software production." — Marc Conrad, Principal Lecturer, University of Bedfordshire, Luton, UK Anyone Can Code: The Art and Science of Logical Creativity introduces computer programming as a way of problem-solving through logical thinking. It uses the notion of modularization as a central lens through which we can make sense of many software concepts. This book takes the reader through fundamental concepts in programming by illustrating them in three different and distinct languages: C/C++, Python, and Javascript. Key features: Focuses on problem-solving and algorithmic thinking instead of programming functions, syntax, and libraries Includes engaging examples, including video games and visual effects Provides exercises and reflective questions This book gives beginner and intermediate learners a strong understanding of what they are doing so that they can do it better and with any other tool or language that they may end up using later. Author Ali Arya is an Associate Professor in the School of Information Technology at Carleton University, Ottawa, Canada. He received his PhD in Computer Engineering from the University of British Columbia, Vancouver, Canada, in 2003. He has over 25 years of experience in professional and academic positions related to software development and information technology. He is passionate about computer programming that brings together logical and creative abilities.

Software -- Software Engineering.

Software architecture—the conceptual glue that holds every phase of a project together for its many stakeholders—is widely recognized as a critical element in modern software development. Practitioners have increasingly discovered that close attention to a software system’s architecture pays valuable dividends. Without an architecture that is appropriate for the problem being solved, a project will stumble along or, most likely, fail. Even with a superb architecture, if that architecture is not well understood or well communicated the project is unlikely to succeed. Documenting Software Architectures, Second Edition, provides the most complete and current guidance, independent of language or notation, on how to capture an architecture in a commonly understandable form. Drawing on their extensive experience, the authors first help you decide what information to document, and then, with guidelines and examples (in various notations, including UML), show you how to express an architecture so that others can successfully build, use, and maintain a system from it. The book features rules for sound documentation, the goals and strategies of documentation, architectural views and styles, documentation for software interfaces and software behavior, and templates for capturing and organizing information to generate a coherent package. New and improved in this second edition: Coverage of architectural styles such as service-oriented architectures, multi-tier architectures, and data models Guidance for documentation in an Agile development environment Deeper treatment of documentation of rationale, reflecting best industrial practices Improved templates, reflecting years of use and feedback, and more documentation layout options A new, comprehensive example (available online), featuring documentation of a Web-based service-oriented system Reference guides for three important architecture documentation languages: UML, AADL, and SySML

New software firms and groups are eager to get their products to market. Even small successes and few customers make these companies grow in size. Resources are usually tight, processes are light, lots of code is written for demos and new people are hired on all the time. In such an environment, it is easy to go astray fast, even for experienced people. This book identifies the things that make for a successful software development organization and provides some solutions and tools to successfully bootstrap a software development group and keep it running efficiently as it grows big. The individual ideas in the article have been tried and tested over many projects and in various companies. The book brings together these ideas in an easy to reference package, provides links to commonly used freeware and inexpensive tools. It also provides some guidelines on how common development practices can be tailored to suit specific needs.

The Art and Science of Logical Creativity

Software Development

FCS Systems Analysis & Design L4

Software Engineering

Domain-driven Design

arc42 by Example

The Best Value Approach to Selecting a Contract Source

Merging the Instructional Design Process with Learner-Centered Theory brings together the innovations of two previously divided processes — learning design strategies/theories and instructional systems development — into a new introductory textbook. Using a holistic rather than fragmented approach that includes top-level, mid-level, and lower-level design, this book provides guidance for major topics such as non-instructional interventions, just-in-time analysis, rapid-prototype approaches, and learner-centered, project-based, anytime-anywhere instruction. Informed by the authors’ considerable experience and leadership throughout dramatic shifts in today’s learning landscape, this book offers the next generation of instructional designers a fresh perspective that synthesizes and pushes beyond the basics of design and development.

There are no easy decisions in software architecture. Instead, there are many hard parts--difficult problems or issues with no best practices--that force you to choose among various compromises. With this book, you'll learn how to think critically about the trade-offs involved with distributed architectures. Architecture veterans and practicing consultants Neal Ford, Mark Richards, Pramod Sadalage, and Zhamak Dehghani discuss strategies for choosing an appropriate architecture. By interweaving a story about a fictional group of technology professionals--the Sysops Squad--they examine everything from how to determine service granularity, manage workflows and orchestration, manage and decouple contracts, and manage distributed transactions to how to optimize operational characteristics, such as scalability, elasticity, and performance. By focusing on commonly asked questions, this book provides techniques to help you discover and weigh the trade-offs as you confront the issues you face as an architect. Analyze trade-offs and effectively document your decisions Make better decisions regarding service granularity Understand the complexities of breaking apart monolithic applications Manage and decouple contracts between services Handle data in a highly distributed architecture Learn patterns to manage workflow and transactions when breaking apart applications

Innovative tools and techniques for the development and design of software systems are essential to the problem solving and planning of software solutions. Software Design and Development: Concepts, Methodologies, Tools, and Applications brings together the best practices of theory and implementation in the development of software systems. This reference source is essential for researchers, engineers, practitioners, and scholars seeking the latest knowledge on the techniques, applications, and methodologies for the design and development of software systems.

An engaging, illustrated collection of insights revealing the practices and principles that expert software designers use to create great software. What makes an expert software designer? It is more than experience or innate ability. Expert software designers have specific habits, learned practices, and observed principles that they apply deliberately during their design work. This book offers sixty-six insights, distilled from years of studying experts at work, that capture what successful software designers actually do to create great software. The book presents these insights in a series of two-page illustrated spreads, with the principle and a short explanatory text on one page, and a drawing on the facing page. For example, “Experts generate alternatives” is illustrated by the same few balloons turned into a set of very different balloon animals. The text is engaging and accessible; the drawings are thought-provoking and often playful. Organized into such categories as “Experts reflect,” “Experts are not afraid,” and “Experts break the rules,” the insights range from “Experts prefer simple solutions” to “Experts see error as opportunity.” Readers learn that “Experts involve the user”; “Experts take inspiration from wherever they can”; “Experts design throughout the creation of software”; and “Experts draw the problem as much as they draw the solution.” One habit for an aspiring expert software designer to develop would be to read and reread this entertaining but essential little book. The insights described offer a guide for the novice or a reference for the veteran—in software design or any design profession. A companion web site provides an annotated bibliography that compiles key underpinning literature, the opportunity to suggest additional insights, and more.

Elements of Reusable Object-Oriented Software

System Engineering Analysis, Design, and Development

Docs Like Code

Software Engineering at Google

JavaPortlet API, Lucene, James, Slide

Playing Autobot Game

Professional Portal Development with Open Source Tools

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, 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

The Information System Consultant's Handbook familiarizes systems analysts, systems designers, and information systems consultants with underlying principles, specific documentation, and methodologies. Corresponding to the primary stages in the systems development life cycle, the book divides into eight sections: Principles Information Gathering and Problem Definition Project Planning and Project Management Systems Analysis Identifying Alternatives Component Design Testing and Implementation Operation and Maintenance Eighty-two chapters comprise the book, and each chapter covers a single tool, technique, set of principles, or methodology. The clear, concise narrative, supplemented with numerous illustrations and diagrams, makes the material accessible for readers - effectively outlining new and unfamiliar analysis and design topics.

With about 200,000 entries, StarBriefs Plus represents the most comprehensive and accurately validated collection of abbreviations, acronyms, contractions and symbols within astronomy, related space sciences and other related fields. As such, this invaluable reference source (and its companion volume, StarGuides Plus) should be on the reference shelf of every library, organization or individual with any interest in these areas. Besides astronomy and associated space sciences, related fields such as aeronautics, aeronomy, astronautics, atmospheric sciences, chemistry, communications, computer sciences, data processing, education, electronics, engineering, energetics, environment, geodesy, geophysics, information handling, management, mathematics, meteorology, optics, physics, remote sensing, and so on, are also covered when justified. Terms in common use and/or of general interest have also been included where appropriate.

This book constitutes the proceedings of the 8th European Conference on Software Architecture, ECSA 2014, held in Vienna, Austria, in August 2014. The 16 full papers and 18 short papers presented in this volume were carefully reviewed and selected from 91 submissions. They are organized in topical sections named: architecture decisions and knowledge; architecture patterns and anti-patterns; reference architectures and metamodels; architecture description languages; enterprise architecture, SOA and cloud computing; components and connectors; quality attributes; and architecture analysis and verification.
Systems Analysis and Design
Software Architecture: The Hard Parts
CONCEPTS AND PRACTICE
Real-Time Systems Design and Analysis
Java 2 Developer
Clinical Engineering Handbook
Managing Software Development
Describes ways to incorporate domain modeling into software development.
"The basic concepts and theories of software engineering have stabilized considerably from the early days of thirty to forty years ago. Nevertheless, the technology and tools continue to evolve, expand and improve every four to five years. In this fifth edition, we will cover some of these newly established improvements in technology and tools but reduce some areas, such as process assessment models, that is becoming less relevant today. We will still maintain many of the historically important concepts that formed the foundation to this field, such as the traditional process models. Our goal is to continue to keep the content of this book to a concise amount that can be taught in a 16-week semester introductory course"--
"An important resource, this book offers an introductory text and overview of real-time systems: systems where timeliness is a crucial part of the correctness of the system. The book contains a pragmatic overview of key topics (computer architecture and organization, operating systems, software engineering, programming languages, and compiler theory) from the perspective of the real-time systems designer. The book is organized into chapters that are essentially self-contained. Thus, the material can be rearranged or omitted depending on the background and interests of the audience or instructor. Each chapter contains both easy and more challenging exercises that stimulate the reader to confront actual problems"--
Looking for a way to invigorate your technical writing team and grow that expertise to include developers, designers, and writers of all backgrounds? When you treat docs like code, you multiply everyoneOs efforts and streamline processes through collaboration, automation, and innovation. Second edition now available with updates and more information about version control for documents and continuous publishing.
The Holistic 4D Model
A Guide for Developers
Designing Secure Software
The Information System Consultant's Handbook
Theory and Practice
Development of N-version Software Samples for an Experiment in Software Fault Tolerance
Documenting Software Architectures

Engineering Software, the third volume in the landmark Write Great Code series by Randall Hyde, helps you create readable and maintainable code that will generate awe from fellow programmers. The field of software engineering may value team productivity over individual growth, but legendary computer scientist Randall Hyde wants to make promising programmers into masters of their craft. To that end, Engineering Software--the latest volume in Hyde's highly regarded Write Great Code series--offers his signature in-depth coverage of everything from development methodologies and strategic productivity to object-oriented design requirements and system documentation. You'll learn: • Why following the software craftsmanship model can lead you to do your best work • How to utilize traceability to enforce consistency within your documentation • The steps for creating your own UML requirements with use-case analysis • How to leverage the IEEE documentation standards to create better software This advanced apprenticeship in the skills, attitudes, and ethics of quality software development reveals the right way to apply engineering principles to programming. Hyde will teach you the rules, and show you when to break them. Along the way, he offers illuminating insights into best practices while empowering you to invent new ones. Brimming with resources and packed with examples, Engineering Software is your go-to guide for writing code that will set you apart from your peers.
Taking a learn-by-doing approach, Software Engineering Design: Theory and Practice uses examples, review questions, chapter exercises, and case study assignments to provide students and practitioners with the understanding required to design complex software systems. Explaining the concepts that are immediately relevant to software designers, it begins with a review of software design fundamentals. The text presents a formal top-down design process that consists of several design activities with varied levels of detail, including the macro-, micro-, and construction-design levels. As part of the top-down approach, it provides in-depth coverage of applied architectural, creational, structural, and behavioral design patterns. For each design issue covered, it includes a step-by-step breakdown of the execution of the design solution, along with an evaluation, discussion, and justification for using that particular solution. The book outlines industry-proven software design practices for leading large-scale software design efforts, developing reusable and high-quality software systems, and producing technical and customer-driven design documentation. It also: Offers one-stop guidance for mastering the Software Design & Construction sections of the official Software Engineering Body of Knowledge (SWEBOK®) Details a collection of standards and guidelines for structuring high-quality code Describes techniques for analyzing and evaluating the quality of software designs Collectively, the text supplies comprehensive coverage of the software design concepts students will need to succeed as professional design leaders. The section on engineering leadership for software designers covers the necessary ethical and leadership skills required of software developers in the public domain. The section on creating software design documents (SDD) familiarizes students with the software design notations, structural descriptions, and behavioral models required for SDDs. Course notes, exercises with answers, online resources, and an instructor's manual are available upon qualified course adoption. Instructors can contact the author about these resources via the author's website: http://softwareengineeringdesign.com/
The nuclear industry and the U.S. Nuclear Regulatory Commission (USNRC) have been working for several years on the development of an adequate process to guide the replacement of aging analog monitoring and control instrumentation in nuclear power plants with modern digital instrumentation without introducing off-setting safety problems. This book identifies criteria for the USNRC's review and acceptance of digital applications in nuclear power plants. It focuses on eight areas: software quality assurance, common-mode software failure potential, systems aspects of digital instrumentation and control technology, human factors and human-machine interfaces, safety and reliability assessment methods, dedication of commercial off-the-shelf hardware and software, the case-by-case licensing process, and the adequacy of technical infrastructure.
Software Development is the most thorough, realistic guide to "what works" in software development - and how to make it happen in your organization. Leading consultant Marc Hamilton tackles all three key elements of successful development: people, processes, and technology. From streamlining infrastructures to retraining programmers, choosing tools to implementing service level agreements, Hamilton unifies all of today's best practices - in management, architecture, and software engineering.
Writing Effective Use Cases
Studies of Software Design
Software Architecture
Software Design and Development: Concepts, Methodologies, Tools, and Applications
Anyone Can Code
ICSE'93 Workshop, Baltimore, Maryland, USA, May (17-18), 1993. Selected Papers
Six Sigma Software Development

Documenting Software ArchitecturesViews and BeyondPearson Education
What every software professional should know about security. Designing Secure Software consolidates Loren Kohnfelder's more than twenty years of experience into a concise, elegant guide to improving the security of technology products. Written for a wide range of software professionals, it emphasizes building security into software design early and involving the entire team in the process. The book begins with a discussion of core concepts like trust, threats, mitigation, secure design patterns, and cryptography. The second part, perhaps this book's most unique and important contribution to the field, covers the process of designing and reviewing a software design with security considerations in mind. The final section details the most common coding flaws that create vulnerabilities, making copious use of code snippets written in C and Python to illustrate implementation vulnerabilities. You'll learn how to: • Identify important assets, the attack surface, and the trust boundaries in a system • Evaluate the effectiveness of various threat mitigation candidates • Work with well-known secure coding patterns and libraries • Understand and prevent vulnerabilities like XSS and CSRF, memory flaws, and more • Use security testing to proactively identify vulnerabilities introduced into code • Review a software design for security flaws effectively and without judgment Kohnfelder's career, spanning decades at Microsoft and Google, introduced numerous software security initiatives, including the co-creation of the STRIDE threat modeling framework used widely today. This book is a modern, pragmatic consolidation of his best practices, insights, and ideas about the future of software.
This textbook aims to prepare students, as well as, practitioners for software design and production. Keeping in mind theory and practice, the book keeps a balance between theoretical foundations and practical considerations. The book by and large meets the requirements of students at all levels of computer science and engineering/information technology for their Software design and Software engineering courses. The book begins with concepts of data and object. This helps in exploring the rationale that guide high level programming language (HLL) design and object oriented frameworks. Once past this post, the book moves on to expand on software design concerns. The book emphasizes the centrality of Parnas's separation of concerns in evolving software designs and architecture. The book extensively explores modelling frameworks such as Unified Modelling Language (UML) and Petri net based methods. Next, the book covers architectural principles and software engineering practices such as Agile – emphasizing software testing during development. It winds up with case studies demonstrating how systems evolve from basic concepts to final products for quality software designs. TARGET AUDIENCE • Undergraduate/postgraduate students of Computer Science and Engineering, and Information Technology • Postgraduate students of Software Engineering/Software Systems
This coherently written book is the final report on the IPSEN project on Integrated Software Project Support Environments devoted to the integration of tools for the development and maintenance of large software systems. The theoretical and application-oriented findings of this comprehensive project are presented in the following chapters: Overview: introduction, classification, and global approach; The outside perspective: tools, environments, their integration, and user interface; Internal conceptual modeling: graph grammar specifications; Realization: derivation of efficient tools, Current and future work, open problems; Conclusion: summary, evaluation, and vision. Also included is a comprehensive bibliography listing more than 1300 entries and a detailed index.
66 Ways Experts Think
A Guide to Best Practices
Design Document Example & Template: Robot Building And Fighting Games
Software architecture documentation in practice
Tools for the Practitioner
Write Great Code, Volume 3
Software Engineering Design
Since Six Sigma has had marked success in improving quality in other settings, and since the quality of software remains poor, it seems a natural evolution to apply the concepts and tools of Six Sigma to system development and the IT department. Until now however, there were no books available that applied these concepts to the system development p
A game design document (GDD) is a software design document that serves as a blueprint from which your game is to be built. It helps you define the scope of your game and sets the general direction for the project, keeping the entire team on the same page.
This is a companion piece, intended to accompany the Lazy Designer series. There are two sections -- a sample planning document followed by a "how it went" discussion on the actual implementation.
Building Tightly Integrated Software Development Environments: The IPSEN Approach
Design Document With A Step By Step Guide: How To Make A Good Game Design Document
Concepts, Methodologies, Tools, and Applications
A Dictionary of Abbreviations, Acronyms and Symbols in Astronomy and Related Space Sciences
Easy Robot Game
Merging the Instructional Design Process with Learner-Centered Theory
Scientific and Technical Aerospace Reports