specification by example tools

specification by example tools are essential components in modern software development, enabling teams to create clear, precise requirements through collaborative examples. These tools facilitate communication between stakeholders, developers, and testers by transforming user stories and acceptance criteria into executable specifications. By integrating examples directly into the development process, specification by example tools help reduce misunderstandings, improve quality, and accelerate delivery. This article explores the most popular specification by example tools, their features, benefits, and how they enhance agile and behavior-driven development (BDD) workflows. Additionally, it covers best practices for choosing and implementing these tools effectively in diverse project environments.

Overview of Specification by Example Tools

Specification by example tools are software applications designed to support the practice of defining requirements through concrete, real-world examples. This approach aligns closely with behavior-driven development (BDD) and acceptance test-driven development (ATDD), emphasizing collaboration and clarity. These tools enable teams to write specifications in a format that is both human-readable and executable, bridging the gap between technical and non-technical stakeholders.

At their core, specification by example tools allow users to create living documentation that evolves alongside the software. This living documentation acts as a single source of truth, ensuring all team members have a shared understanding of the product's expected behavior. The tools support writing scenarios that describe how features should behave under various conditions, often in a structured format such as Gherkin or similar domain-specific languages.

Purpose and Functionality

The primary purpose of specification by example tools is to capture requirements as concrete examples that can be validated automatically or manually. These examples serve multiple functions:

- Clarify ambiguous requirements by providing concrete scenarios
- Facilitate automated testing through executable specifications

- Improve collaboration among product owners, developers, and testers
- Maintain traceability between requirements and tests

By embedding examples directly into the development lifecycle, these tools help reduce defects, streamline communication, and accelerate feedback loops.

Key Features of Specification by Example Tools

Specification by example tools offer a range of features tailored to enhance requirements management and test automation. Understanding these features is critical when selecting the right tool for a project.

Collaborative Authoring

Most specification by example tools provide collaborative environments where stakeholders can co-author and review specifications. This functionality often includes version control, commenting, and role-based access to ensure transparency and accountability.

Executable Specifications

One of the defining features is the ability to convert written examples into executable tests. These tools integrate with testing frameworks and continuous integration (CI) pipelines to automate acceptance testing, reducing manual effort and increasing reliability.

Traceability and Reporting

Effective traceability features link specifications to corresponding tests, code, and defects. Comprehensive reporting capabilities enable teams to monitor test coverage, identify gaps, and track progress against requirements.

Integration with Development Ecosystem

Leading specification by example tools seamlessly integrate with popular development environments, issue trackers, and CI/CD tools. This interoperability facilitates smooth workflows and enhances productivity.

Support for Domain-Specific Languages

These tools often support domain-specific languages (DSLs) such as Gherkin, which allow writing scenarios in a natural language style. This makes specifications accessible to both technical and business users.

Popular Specification by Example Tools in the Market

Several specification by example tools have gained prominence due to their robust features, community support, and adaptability. The following are among the most widely used in the industry.

Cucumber

Cucumber is a widely adopted open-source tool that supports BDD by allowing users to write executable specifications in Gherkin syntax. It integrates with multiple programming languages and testing frameworks, making it versatile for various project types.

SpecFlow

SpecFlow is the .NET equivalent of Cucumber, designed to enable BDD practices in Microsoft environments. It supports Gherkin syntax and integrates well with Visual Studio, Azure DevOps, and other Microsoft tools.

LivingDoc

LivingDoc focuses on generating living documentation from specification by example artifacts. It provides an intuitive interface for reviewing and maintaining up-to-date specifications linked to automated tests.

FitNesse

FitNesse is a wiki-based tool that allows users to create and run acceptance tests as living documentation. It emphasizes collaboration and is suitable for teams that value a knowledge-sharing culture.

Behave

Behave is a BDD framework for Python that supports writing specifications in Gherkin format. It facilitates integration of specification by example practices within Python development workflows.

Benefits of Using Specification by Example Tools

Incorporating specification by example tools into software development processes delivers numerous advantages. These benefits contribute to improved product quality and team efficiency.

Enhanced Communication

By using concrete examples as a common language, specification by example tools bridge the gap between business and technical teams. This reduces misunderstandings and ensures that all stakeholders share the same expectations.

Improved Quality and Reduced Defects

Executable specifications serve as automated acceptance tests that validate functionality continuously. This early and ongoing validation helps detect defects sooner, reducing costly rework.

Faster Feedback Loops

Automation enabled by these tools accelerates feedback to developers and product owners, allowing rapid iteration and continuous improvement.

Living Documentation

The specifications created remain current and relevant as the software evolves, eliminating the problem of outdated or forgotten documentation.

Traceability and Compliance

Specification by example tools provide traceability from requirements to tests and defects, supporting regulatory compliance and audit readiness.

Implementing Specification by Example Tools in Agile Environments

Agile methodologies emphasize collaboration, flexibility, and rapid delivery, making specification by example tools an ideal fit. Implementing these tools effectively requires alignment with agile principles and practices.

Integration with Agile Ceremonies

Specification by example tools facilitate discussions during sprint planning, backlog refinement, and retrospectives. Using concrete examples helps teams clarify requirements early and adjust based on feedback.

Continuous Integration and Delivery

By integrating specifications with CI/CD pipelines, teams ensure that acceptance tests run automatically with every build. This supports the agile goal of delivering potentially shippable increments frequently.

Collaborative Specification Workshops

Workshops involving product owners, developers, and testers use specification by example tools to create and refine examples collaboratively. This practice fosters shared understanding and ownership.

Best Practices for Effective Use of Specification by Example Tools

To maximize the benefits of specification by example tools, organizations should consider the following best practices.

1. **Start with Clear, Understandable Examples:** Ensure that examples are simple, relevant, and focused on business value.

2. **Promote Cross-Functional Collaboration:** Involve all stakeholders in defining and reviewing specifications to enhance clarity and buy-in.

3. **Automate Acceptance Tests:** Link specifications to automated tests to enable continuous

validation and faster feedback.

4. **Maintain Living Documentation:** Regularly update and review specifications to keep them aligned with evolving requirements.

5. **Train Teams on Tool Usage:** Provide adequate training and support to ensure effective adoption and consistent use.

6. **Integrate with Existing Toolchains:** Leverage integrations with issue trackers, CI/CD, and development environments to streamline workflows.

Adhering to these practices helps organizations fully leverage specification by example tools to enhance software quality, collaboration, and delivery speed.

Frequently Asked Questions

What is Specification by Example (SBE) in software development?

Specification by Example is a collaborative approach to defining requirements and functional tests using realistic examples, ensuring shared understanding among stakeholders and improving software quality.

What are the key benefits of using Specification by Example tools?

Specification by Example tools help improve communication between developers, testers, and business stakeholders, reduce misunderstandings, enable living documentation, and automate acceptance testing based on real-world examples.

Which are some popular Specification by Example tools available in the market?

Popular Specification by Example tools include Cucumber, SpecFlow, FitNesse, Concordion, and LivingDoc, among others.

How do Specification by Example tools integrate with Agile development processes?

Specification by Example tools support Agile by facilitating collaboration, enabling continuous feedback through automated acceptance tests, and maintaining up-to-date living documentation that evolves with the product.

Can Specification by Example tools be used with Behavior-Driven Development (BDD)?

Yes, Specification by Example tools are often used alongside BDD practices since both emphasize collaboration, shared understanding, and executable specifications written in natural language.

What formats do Specification by Example tools typically use for defining examples?

Most Specification by Example tools use structured natural language formats like Gherkin, which allow writing scenarios in Given-When-Then syntax that are easy for both technical and non-technical stakeholders to understand.

How do Specification by Example tools help in automating acceptance testing?

These tools link the examples defined in specifications directly to automated test scripts, enabling tests to be executed automatically and ensuring the application meets the specified requirements continuously.

Are Specification by Example tools suitable for non-technical stakeholders?

Yes, Specification by Example tools are designed to be readable and writable by non-technical stakeholders, promoting collaboration and shared understanding across business and technical teams.

What challenges might teams face when adopting Specification by Example tools?

Challenges include the initial learning curve, maintaining examples as requirements change, ensuring collaboration among diverse teams, and integrating tools into existing development workflows.

How can teams measure the effectiveness of Specification by Example tools?

Teams can measure effectiveness by tracking improved communication, reduced defects, faster feedback cycles, increased automation of acceptance tests, and higher alignment between delivered software and business requirements.

Additional Resources

1. *Specification by Example: How Successful Teams Deliver the Right Software*
This book, authored by Gojko Adzic, is a foundational guide to Specification by Example (SBE). It explains how teams can use realistic examples to define requirements and automate functional tests, ensuring that the delivered software meets business needs. The book covers practical techniques,

real-world case studies, and best practices for collaboration between developers, testers, and business stakeholders.

2. *Bridging the Communication Gap: Specification by Example and Agile Acceptance Testing*
Written by Gojko Adzic, this book focuses on improving communication between technical teams and business stakeholders through Specification by Example. It provides actionable advice on writing clear, executable requirements and acceptance tests that serve as living documentation. The book is ideal for teams adopting Agile methodologies and aiming to reduce misunderstandings and rework.

3. *Living Documentation: Continuous Documentation with Specification by Example*
This book delves into the concept of living documentation, where specifications are continuously updated and validated through automated tests. It highlights tools and techniques that integrate with Specification by Example to keep documentation accurate and relevant throughout the software development lifecycle. Readers will gain insights into maintaining clear and up-to-date documentation that drives collaboration.

4. *BDD in Action: Behavior-Driven Development for the Whole Software Lifecycle*
Though focused on Behavior-Driven Development (BDD), this book covers many principles overlapping with Specification by Example. It guides readers through creating executable specifications using tools like Cucumber, SpecFlow, and JBehave. The book emphasizes collaboration, automation, and delivering software that meets business expectations.

5. *Agile Testing: A Practical Guide for Testers and Agile Teams*
By Lisa Crispin and Janet Gregory, this comprehensive guide includes coverage of Specification by Example techniques as part of Agile testing practices. It teaches how testers can collaborate with developers and product owners to define clear acceptance criteria and automate tests. The book offers strategies to improve quality and ensure that software delivers value.

6. *Example Mapping: Bridging the Gap Between User Stories and Acceptance Tests*
This book introduces Example Mapping as a facilitation technique to help teams break down user stories into concrete examples and rules. It complements Specification by Example by providing a structured way to gather and organize requirements collaboratively. The approach improves clarity and helps avoid ambiguity in acceptance criteria.

7. *Test Automation with SpecFlow: Bridging the Gap Between Business and Technical Teams*
Focused on the SpecFlow tool, this book demonstrates how to implement Specification by Example in .NET environments. It covers writing Gherkin-based scenarios, automating acceptance tests, and integrating with continuous integration pipelines. The book is practical for teams looking to adopt BDD and SBE practices using SpecFlow.

8. *Collaborative Specification: Techniques for Agile Teams*
This book explores various collaborative techniques, including Specification by Example, to enhance communication and shared understanding among Agile teams. It offers practical methods for gathering requirements, defining acceptance criteria, and automating tests. The focus is on fostering teamwork to deliver high-quality software efficiently.

9. *Effective Acceptance Test-Driven Development with Agile Principles*
This book combines principles of acceptance test-driven development (ATDD) with Specification by Example practices. It guides readers through creating acceptance tests that serve as specifications and automated checks, ensuring software aligns with customer needs. The book includes real-life examples and emphasizes continuous collaboration between stakeholders.

Specification By Example Tools

specification by example tools: Specification by Example Gojko Adzic, 2011-06-02 Summary Specification by Example is an emerging practice for creating software based on realistic examples, bridging the communication gap between business stakeholders and the dev teams building the software. In this book, author Gojko Adzic distills interviews with successful teams worldwide, sharing how they specify, develop, and deliver software, without defects, in short iterative delivery cycles. About the Technology Specification by Example is a collaborative method for specifying requirements and tests. Seven patterns, fully explored in this book, are key to making the method effective. The method has four main benefits: it produces living, reliable documentation; it defines expectations clearly and makes validation efficient; it reduces rework; and, above all, it assures delivery teams and business stakeholders that the software that's built is right for its purpose. About the Book This book distills from the experience of leading teams worldwide effective ways to specify, test, and deliver software in short, iterative delivery cycles. Case studies in this book range from small web startups to large financial institutions, working in many processes including XP, Scrum, and Kanban. This book is written for developers, testers, analysts, and business people working together to build great software. Purchase of the print book comes with an offer of a free PDF, ePub, and Kindle eBook from Manning. Also available is all code from the book. What's Inside Common process patterns How to avoid bad practices Fitting SBE in your process 50+ case studies ========== Table of Contents Part 1 Getting started Part 2 Key process patterns Part 3 Case studies Key benefits Key process patterns Living documentation Initiating the changes Deriving scope from goals Specifying collaboratively Illustrating using examples Refining the specification Automating validation without changing specifications Validating frequently Evolving a documentation system uSwitch RainStor Iowa Student Loan Sabre Airline Solutions ePlan Services Songkick Concluding thoughts

specification by example tools: ATDD by Example Markus Gärtner, 2013 With Acceptance Test-Driven Development (ATDD), business customers, testers, and developers can collaborate to produce testable requirements that help them build higher quality software more rapidly. However, ATDD is still widely misunderstood by many practitioners. ATDD by Example is the first practical, entry-level, hands-on guide to implementing and successfully applying it. ATDD pioneer Markus Gärtner walks readers step by step through deriving the right systems from business users, and then implementing fully automated, functional tests that accurately reflect business requirements, are intelligible to stakeholders, and promote more effective development. Through two end-to-end case studies, Gärtner demonstrates how ATDD can be applied using diverse frameworks and languages. Each case study is accompanied by an extensive set of artifacts, including test automation classes, step definitions, and full sample implementations. These realistic examples illuminate ATDD's fundamental principles, show how ATDD fits into the broader development process, highlight tips from Gärtner's extensive experience, and identify crucial pitfalls to avoid. Readers will learn to Master the thought processes associated with successful ATDD implementation Use ATDD with Cucumber to describe software in ways businesspeople can understand Test web pages using ATDD tools Bring ATDD to Java with the FitNesse wiki-based acceptance test framework Use examples more effectively in Behavior-Driven Development (BDD) Specify software collaboratively through

innovative workshops Implement more user-friendly and collaborative test automation Test more cleanly, listen to test results, and refactor tests for greater value If you're a tester, analyst, developer, or project manager, this book offers a concrete foundation for achieving real benefits with ATDD now-and it will help you reap even more value as you gain experience.

specification by example tools: Agile Processes in Software Engineering and Extreme Programming Giovanni Cantone, Michele Marchesi, 2014-06-30 This book contains the refereed proceedings of the 15th International Conference on Agile Software Development, XP 2014, held in Rome, Italy, in May 2014. Because of the wide application of agile approaches in industry, the need for collaboration between academics and practitioners has increased in order to develop the body of knowledge available to support managers, system engineers, and software engineers in their managerial/economic and architectural/project/technical decisions. Year after year, the XP conference has facilitated such improvements and provided evidence on the advantages of agile methodologies by examining the latest theories, practical applications, and implications of agile and lean methods. The 15 full papers, seven short papers, and four experience reports accepted for XP 2014 were selected from 59 submissions and are organized in sections on: agile development, agile challenges and contracting, lessons learned and agile maturity, how to evolve software engineering teaching, methods and metrics, and lean development.

specification by example tools: *CESAR - Cost-efficient Methods and Processes for Safety-relevant Embedded Systems* Ajitha Rajan, Thomas Wahl, 2013-03-25 The book summarizes the findings and contributions of the European ARTEMIS project, CESAR, for improving and enabling interoperability of methods, tools, and processes to meet the demands in embedded systems development across four domains - avionics, automotive, automation, and rail. The contributions give insight to an improved engineering and safety process life-cycle for the development of safety critical systems. They present new concept of engineering tools integration platform to improve the development of safety critical embedded systems and illustrate capacity of this framework for end-user instantiation to specific domain needs and processes. They also advance state-of-the-art in component-based development as well as component and system validation and verification, with tool support. And finally they describe industry relevant evaluated processes and methods especially designed for the embedded systems sector as well as easy adoptable common interoperability principles for software tool integration.

specification by example tools: Test Driven Lasse Koskela, 2007-08-31 In test driven development, you first write an executable test ofwhat your application code must do. Only then do you write thecode itself and, with the test spurring you on, you improve yourdesign. In acceptance test driven development (ATDD), you usethe same technique to implement product features, benefiting fromiterative development, rapid feedback cycles, and better-definedrequirements. TDD and its supporting tools and techniques leadto better software faster. Test Driven brings under one cover practical TDD techniquesdistilled from several years of community experience. With examplesin Java and the Java EE environment, it explores both the techniquesand the mindset of TDD and ATDD. It uses carefully chosen examplesto illustrate TDD tools and design patterns, not in the abstractbut concretely in the context of the technologies you face at work.It is accessible to TDD beginners, and it offers effective and less wellknown techniques to older TDD hands. Purchase of the print book comes with an offer of a free PDF, ePub, and Kindle eBook from Manning. Also available is all code from the book. What's Inside Learn hands-on to test drive Java code How to avoid common TDD adoption pitfalls Acceptance test driven development and the Fit framework How to test Java EE components-Servlets, JSPs, and SpringControllers Tough issues like multithreaded programs and data access code

specification by example tools: Verified Software: Theories, Tools, Experiments Bertrand Meyer, Jim Woodcock, 2008-06-29 A Step Towards Verified Software Worries about the reliability of software are as old as software itself; techniques for allaying these worries predate even James King’s 1969 thesis on “A program verifier. ” What gives the whole topic a new urgency is the conjunction of three phenomena: the blitz-like spread of software-rich systems to control ever more

facets of our world and our lives; our growing impatience with deficiencies; and the development—proceeding more slowly, alas, than the other two trends—of techniques to ensure and verify software quality. In 2002 Tony Hoare, one of the most distinguished contributors to these advances over the past four decades, came to the conclusion that piecemeal efforts are no longer sufficient and proposed a “Grand Challenge” intended to achieve, over 15 years, the production of a verifying compiler: a tool that while processing programs would also guarantee their adherence to specified properties of correctness, robustness, safety, security and other desirable properties. As Hoare sees it, this endeavor is not a mere research project, as might normally be carried out by one team or a small consortium of teams, but a momentous endeavor, comparable in its scope to the successful mission to send a man to the moon or to the sequencing of the human genome.

specification by example tools: The Art of Agile Development James Shore, Shane Warden, 2021-10-12 Most companies developing software employ something they call Agile. But there's widespread misunderstanding of what Agile is and how to use it. If you want to improve your software development team's agility, this comprehensive guidebook's clear, concrete, and detailed guidance explains what to do and why, and when to make trade-offs. In this thorough update of the classic Agile how-to guide, James Shore provides no-nonsense advice on Agile adoption, planning, development, delivery, and management taken from over two decades of Agile experience. He brings the latest ideas from Extreme Programming, Scrum, Lean, DevOps, and more into a cohesive whole. Learn how to successfully bring Agile development to your team and organization--or discover why Agile might not be for you. This book explains how to: Improve agility: create the conditions necessary for Agile to succeed and scale in your organization Focus on value: work as a team, understand priorities, provide visibility, and improve continuously Deliver software reliably: share ownership, decrease development costs, evolve designs, and deploy continuously Optimize value: take ownership of product plans, budgets, and experiments--and produce market-leading software

specification by example tools: Conceptual Modeling for E-Business and the Web Stephen W. Liddle, Heinrich C. Mayr, Bernhard Thalheim, 2003-06-29 The objective of the workshops associated with the ER2000 19th International Conference on Conceptual Modeling was to give participants the opportunity to present and discuss emerging, hot topics, thus adding new perspectives to conceptual modeling. This attracts communities which have begun to or which have already recognized the importance of conceptual modeling for solving their problems. To meet this objective, we selected the following two topics: { Conceptual Modeling Approaches for E-Business (eCOMO2000) aimed at studying the application of conceptual modeling techniques speci cally to e-business. { The World Wide Web and Conceptual Modeling (WCM2000) which analyzes how conceptual modeling can help address the challenges of Web devel- ment, management, and use. eCOMO2000 is the rst international workshop on Conceptual Modeling - proaches for E-Business. It was intended to work out and to discuss the actual state of research on conceptual modeling aspects and methods within the realm of the network economy, which is driven by both traditionally organized ent- prises and dynamic networks. Following the philosophy of the ER workshops, the selection of eCOMO contributions was done very carefully and restrictively (six accepted papers out of thirteen submissions) in order to guarantee an excellent workshop program. We are deeply indebted to the authors and to the members of the program committee, whose work resulted in this outstanding program.

specification by example tools: Model Driven Engineering Languages and Systems Andy Schürr, Bran V. Selic, 2009-09-30 The pioneering organizers of the ?rst UML workshop in Mulhouse, France inthe summerof1998couldhardlyhaveanticipatedthat,in littleoveradecade, theirinitiativewouldblossomintotoday’shighlysuccessfulMODELSconference series, the premier annual gathering of researchersand practitioners focusing on a very important new technical discipline: model-based software and system engineering. This expansion is, of course, a direct consequence of the growing signi?cance and success of model-based methods in practice. The conferences have contributed greatly to the heightened interest in the ?eld, attracting much young talent and leading to the gradualemergence of its correspondingscienti?c and engineering

foundations. The proceedings from the MODELS conferences are one of the primary references for anyone interested in a more substantive study of the domain. The 12th conference took place in Denver in the USA, October 4–9, 2009 along with numerous satellite workshops and tutorials, as well as several other related scienti?c gatherings. The conference was exceptionally fortunate to have three eminent, invited keynote speakers from industry: Stephen Mellor, Larry Constantine, and Grady Booch.

specification by example tools: *BDD in Action, Second Edition* John Ferguson Smart, Jan Molak, 2023-06-20 Deliver software that does what it's supposed to do! Behavior-Driven Development guides your software projects to success with collaboration, communication techniques, and concrete requirements you can turn into automated tests. In BDD in Action, Second Edition you'll learn how to: Implement and improve BDD practices Prioritize features from business goals Facilitate an example mapping session Write automated acceptance tests Scale up your automated acceptance tests Deliver accurate reporting and documentation Around half of all software projects fail to deliver on requirements. Behavior-Driven Development (BDD) helps make sure that yours isn't one of them. Behavior-Driven Development in Action, Second Edition teaches you how to ensure that everyone involved in a software project—from developers to non-technical stakeholders—are in agreement on goals and objectives. It lays out the communication skills, collaborative practices, and useful automation tools that will let you seamlessly succeed with BDD. Now in its second edition, this revised bestseller has been extensively updated with new techniques for incorporating BDD into large-scale and enterprise development practices such as Agile and DevOps. Foreword by Daniel Terhorst-North. About the Technology Behavior-Driven Development is a collaborative software design technique that organizes examples of an application's desired behavior into a concrete, testable specification. Because the BDD process gathers input from all areas of an organization, it maximizes the likelihood your software will satisfy both end users and business stakeholders. The established collaboration practices and automation strategies in this book will help you maximize the benefits of BDD for your dev team and your business clients. About the Book In BDD in Action, Second Edition, you'll learn to seamlessly integrate BDD into your existing development process. This thoroughly revised new edition now shows how to integrate BDD with DevOps and large-scale Agile systems. Practical examples introduce cross-functional team communication skills, leading a successful requirements analysis, and how to set up automated acceptance criteria. What's Inside How BDD positively affects teamwork, dynamics, and collaboration with stakeholders Help teams discover and analyze requirements, uncover assumptions, and reduce risks Make acceptance, integration, and unit testing more effective Automate reporting and living documentation to improve transparency About the Reader For all development teams. No experience with BDD required. Examples in Java, JavaScript, and TypeScript can be easily expressed in your chosen language. About the Author John Ferguson Smart is the creator of the Serenity BDD framework and founder of the Serenity Dojo training school. Jan Molak is the author of the Serenity/JS testing framework, Jenkins Build Monitor, and other CD and testing tools.

specification by example tools: KORSO: Methods, Languages, and Tools for the Construction of Correct Software Manfred Broy, Stefan Jähnichen, 1995-11-08 This book constitutes the final report of the work carried out in the project KORSO (Korrekte Software) funded by the German Federal Ministry for Research and Technology. KORSO is an evolutionary, prototype-oriented project aimed at improving the theoretical foundations of quality-driven software engineering and at implementing known techniques for applications of practical relevance. The 21 strictly refereed papers presented are organized in five sections on methods for correctness, languages, development systems and logical frameworks, tools, and case studies. In addition, the preface and introductory paper give valuable background information and a concise state-of-the-art overview.

specification by example tools: Analysis and Visualization Tools for Constraint Programming Pierre Deransart, M.V. Hermenegildo, J. Maluszynski, 2006-12-31 Coordinating

production across a supply chain, designing a new VLSI chip, allocating classrooms or scheduling maintenance crews at an airport are just a few examples of complex (combinatorial) problems that can be modeled as a set of decision variables whose values are subject to a set of constraints. The decision variables may be the time when production of a particular lot will start or the plane that a maintenance crew will be working on at a given time. Constraints may range from the number of students you can ?t in a given classroom to the time it takes to transfer a lot from one plant to another.Despiteadvancesincomputingpower,manyformsoftheseandother combinatorial problems have continued to defy conventional programming approaches. Constraint Logic Programming (CLP) ?rst emerged in the mid-eighties as a programming technique with the potential of signi?cantly reducing the time it takes to develop practical solutions to many of these problems, by combining the expressiveness of languages such as Prolog with the compu- tional power of constrained search. While the roots of CLP can be traced to Monash University in Australia, it is without any doubt in Europe that this new software technology has gained the most prominence, bene?ting, among other things, from sustained funding from both industry and public R&D programs over the past dozen years. These investments have already paid o?, resulting in a number of popular commercial solutions as well as the creation of several successful European startups.

specification by example tools: *Proceedings of The International Conference on eBusiness, eCommerce, eManagement, eLearning and eGovernance 2014* Maaruf Ali, Mahdi H. Miraz, Kokula Krishna Hari Kunasekaran, 2013-07-19 International Conference on eBusiness, eCommerce, eManagement, eLearning and eGovernance 2014 University of Greenwich, London, England

specification by example tools: Writing Great Specifications Kamil Nicieja, 2017-10-25 Summary Writing Great Specifications is an example-rich tutorial that teaches you how to write good Gherkin specification documents that take advantage of the benefits of specification by example. Foreword written by Gojko Adzic. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About the Technology The clearest way to communicate a software specification is to provide examples of how it should work. Turning these story-based descriptions into a well-organized dev plan is another matter. Gherkin is a human-friendly, jargon-free language for documenting a suite of examples as an executable specification. It fosters efficient collaboration between business and dev teams, and it's an excellent foundation for the specification by example (SBE) process. About the Book Writing Great Specifications teaches you how to capture executable software designs in Gherkin following the SBE method. Written for both developers and non-technical team members, this practical book starts with collecting individual feature stories and organizing them into a full, testable spec. You'll learn to choose the best scenarios, write them in a way that anyone can understand, and ensure they can be easily updated by anyone.management. What's Inside Reading and writing Gherkin Designing story-based test cases Team Collaboration Managing a suite of Gherkin documents About the Reader Primarily written for developers and architects, this book is accessible to any member of a software design team. About the Author Kamil Nicieja is a seasoned engineer, architect, and project manager with deep expertise in Gherkin and SBE. Table of contents Introduction to specification by example and Gherkin PART 1 - WRITING EXECUTABLE SPECIFICATIONS WITH EXAMPLES The specification layer and the automation layer Mastering the Given-When-Then template The basics of scenario outlines Choosing examples for scenario outlines The life cycle of executable specifications Living documentation PART 2 - MANAGING SPECIFICATION SUITES Organizing scenarios into a specification suite Refactoring features into abilities and business needs Building a domain-driven specification suite Managing large projects with bounded contexts

specification by example tools: *Succeeding with Agile* Mike Cohn, 2010 Proven, 100% Practical Guidance for Making Scrum and Agile Work in Any Organization This is the definitive, realistic, actionable guide to starting fast with Scrum and agile-and then succeeding over the long haul. Leading agile consultant and practitioner Mike Cohn presents detailed recommendations, powerful tips, and real-world case studies drawn from his unparalleled experience helping hundreds of software organizations make Scrum and agile work. Succeeding with Agile is for pragmatic

software professionals who want real answers to the most difficult challenges they face in implementing Scrum. Cohn covers every facet of the transition: getting started, helping individuals transition to new roles, structuring teams, scaling up, working with a distributed team, and finally, implementing effective metrics and continuous improvement. Throughout, Cohn presents Things to Try Now sections based on his most successful advice. Complementary Objection sections reproduce typical conversations with those resisting change and offer practical guidance for addressing their concerns. Coverage includes Practical ways to get started immediately-and get good fast Overcoming individual resistance to the changes Scrum requires Staffing Scrum projects and building effective teams Establishing improvement communities of people who are passionate about driving change Choosing which agile technical practices to use or experiment with Leading self-organizing teams Making the most of Scrum sprints, planning, and quality techniques Scaling Scrum to distributed, multiteam projects Using Scrum on projects with complex sequential processes or challenging compliance and governance requirements Understanding Scrum's impact on HR, facilities, and project management Whether you've completed a few sprints or multiple agile projects and whatever your role-manager, developer, coach, ScrumMaster, product owner, analyst, team lead, or project lead-this book will help you succeed with your very next project. Then, it will help you go much further: It will help you transform your entire development organization.

specification by example tools: Semantic Web Services: Theory, Tools and Applications Cardoso, Jorge, 2007-03-31 This book brings together researchers, scientists, and representatives from different communities to study, understand, and explore the theory, tools, and applications of the semantic Web. It joins the semantic Web, ontologies, knowledge management, Web services, and Web processes into one fully comprehensive resource, serving as the platform for exchange of both practical technologies and research--Provided by publisher.

specification by example tools: Field Device Tool - FDT Rene Simon, 2005 This book describes the processes and technologies for embedding field devices, from the perspective of the various automation applications and from the perspective of the devices, and reveals the similarities. It provides a detailed explanation of the essential components and processes, such as instantiation, commissioning and channel assignment. It also details the architecture concepts of DTMs for communication connection devices and remote I/Os. An introduction to the FDT style guide describes the interface between the end user programmer. This title is oriented equally towards corporate decision-makers, developers industrial automation companies who provide devices and systems, and system integrators. Readers will be able to gain an appreciation for the importance of FDT technology for products, to initiate DTM developments and to integrate FDT-based components into systems. This book is based upon Version 1.2 of the FDT specification and its addendum.

specification by example tools: Human-computer Interaction Jenny Preece, Laurie S. Keller, 1990 01 $aZie ook de Ou-cursus: Human-computer interaction. Zie ook de Ou-cursus: Human-computer interaction.

specification by example tools: Agile Software Development - An Overview K Amuthabala, Shantala Devi Patil, Thirumagal E, Thanuja K, 2023-10-05 This textbook has been meticulously crafted with a singular purpose: offering a comprehensive and practical guide to Agile Software Development. In the forthcoming chapters, we will delve into theintricacies of Agile methodologies, explore their underlying principles, and investigate the compelling reasons behind their prominence in the software development industry. Section I: Introduction to Iterative Development, Evolutionary, and Adaptive Development, Our journeybegins with an exploration of fundamental concepts: Iterative Development, Evolutionary Development,and Adaptive Development. These approaches break free from conventional linear development processesand prioritize flexibility, risk management, and client-driven planning. This chapter will discuss the meritsof time-boxed iterative development, evolutionary requirements analysis, incremental delivery, and theultimate goal of evolutionary delivery. Section II: Serves as a bridge between theory and practice within the Agile realm. Here, we define AgileDevelopment, categorize various methodologies, and delve deep into the Agile Manifesto and its guidingprinciples. Additionally, we explore Agile project management,

emphasizing the crucial role ofcommunication, feedback, and the human element. The chapter culminates in an exploration of specificAgile methods and a balanced discussion of the ongoing discourse surrounding Agile Hype. Section III: Motivation and Evidence, Understanding the motivation underpinning Agile is fundamental toappreciating its significance. In Chapter 3, we illuminate the imperatives for change in software projectsand how iterative development addresses these challenges. We critique the limitations of the traditionalWaterfall model and provide a comprehensive review of supporting evidence, including research findings,historical project data, and expert opinions, all converging to fortify the case for iterative development. Section IV: Fundamentals of DevOps and Technical View, Agile methodologies extend beyond softwaredevelopment into the realm of DevOps. Chapter 4 introduces the foundational principles of DevOps and itspivotal role in contemporary development practices. We delve into the building blocks of DevOps, thevital metrics and measurement perspective, and the process view that fosters seamless collaborationbetween development and operations teams. The section IV concludes with an in-depth exploration of thetechnical facets, including topics like automatic releasing, infrastructure as code, and specification byexample, enriched by real-world case studies. Upon completing this textbook, you will comprehensively comprehend Agile Software Development andDevOps. Whether you are a student embarking on a career in software development or an industryprofessional looking to stay at the forefront of the field, the knowledge and insights provided here will equip you with the tools to excel in the dynamic world of software development. Let us embark on this enlightening journey together, embracing agility, adaptability, and excellence in software development.

specification by example tools: The Future of Software Quality Assurance Stephan Goericke, 2019-11-19 This open access book, published to mark the 15th anniversary of the International Software Quality Institute (iSQI), is intended to raise the profile of software testers and their profession. It gathers contributions by respected software testing experts in order to highlight the state of the art as well as future challenges and trends. In addition, it covers current and emerging technologies like test automation, DevOps, and artificial intelligence methodologies used for software testing, before taking a look into the future. The contributing authors answer questions like: How is the profession of tester currently changing? What should testers be prepared for in the years to come, and what skills will the next generation need? What opportunities are available for further training today? What will testing look like in an agile world that is user-centered and fast-paced? What tasks will remain for testers once the most important processes are automated? iSQI has been focused on the education and certification of software testers for fifteen years now, and in the process has contributed to improving the quality of software in many areas. The papers gathered here clearly reflect the numerous ways in which software quality assurance can play a critical role in various areas. Accordingly, the book will be of interest to both professional software testers and managers working in software testing or software quality assurance.

Related to specification by example tools

SPECIFICATION Definition & Meaning - Merriam-Webster The meaning of SPECIFICATION is the act or process of specifying. How to use specification in a sentence

Specification (technical standard) - Wikipedia Although specifications are usually issued by the architect 's office, specification writing itself is undertaken by the architect and the various engineers or by specialist specification writers

SPECIFICATION | English meaning - Cambridge Dictionary SPECIFICATION definition: 1. a detailed description of how something should be done, made, etc.: 2. a detailed description. Learn more

specification noun - Definition, pictures, pronunciation and usage Definition of specification noun in Oxford Advanced Learner's Dictionary. Meaning, pronunciation, picture, example sentences, grammar, usage notes, synonyms and more

Specification Definition & Meaning | Britannica Dictionary SPECIFICATION meaning: a detailed description of work to be done or materials to be used in a project an instruction that says

exactly how to do or make something usually plural

Specification - definition of specification by The Free Dictionary specification (ˌspɛsɪfɪˈkeɪʃən) n 1. the act or an instance of specifying

SPECIFICATION - Definition & Translations | Collins English Discover everything about the word "SPECIFICATION" in English: meanings, translations, synonyms, pronunciations, examples, and grammar insights - all in one comprehensive guide

specification - Dictionary of English the act of specifying:[uncountable] specification of charges against the prisoner. Usually, specifications. [plural] a detailed description of requirements, materials, etc., as in a plan for a

SPECIFICATION Definition & Meaning | Specification definition: the act of specifying.. See examples of SPECIFICATION used in a sentence

What is a Specification? - Specright The purpose of a specification is to manage the development process via detailed descriptions of the product design that represent the opinion of what a company wants

SPECIFICATION Definition & Meaning - Merriam-Webster The meaning of SPECIFICATION is the act or process of specifying. How to use specification in a sentence

Specification (technical standard) - Wikipedia Although specifications are usually issued by the architect 's office, specification writing itself is undertaken by the architect and the various engineers or by specialist specification writers

SPECIFICATION | English meaning - Cambridge Dictionary SPECIFICATION definition: 1. a detailed description of how something should be done, made, etc.: 2. a detailed description. Learn more

specification noun - Definition, pictures, pronunciation and usage Definition of specification noun in Oxford Advanced Learner's Dictionary. Meaning, pronunciation, picture, example sentences, grammar, usage notes, synonyms and more

Specification Definition & Meaning | Britannica Dictionary SPECIFICATION meaning: a detailed description of work to be done or materials to be used in a project an instruction that says exactly how to do or make something usually plural

Specification - definition of specification by The Free Dictionary specification (ˌspɛsɪfɪˈkeɪʃən) n 1. the act or an instance of specifying

SPECIFICATION - Definition & Translations | Collins English Discover everything about the word "SPECIFICATION" in English: meanings, translations, synonyms, pronunciations, examples, and grammar insights - all in one comprehensive guide

specification - Dictionary of English the act of specifying:[uncountable] specification of charges against the prisoner. Usually, specifications. [plural] a detailed description of requirements, materials, etc., as in a plan for a

SPECIFICATION Definition & Meaning | Specification definition: the act of specifying.. See examples of SPECIFICATION used in a sentence

What is a Specification? - Specright The purpose of a specification is to manage the development process via detailed descriptions of the product design that represent the opinion of what a company wants

Back to Home: https://explore.gcts.edu