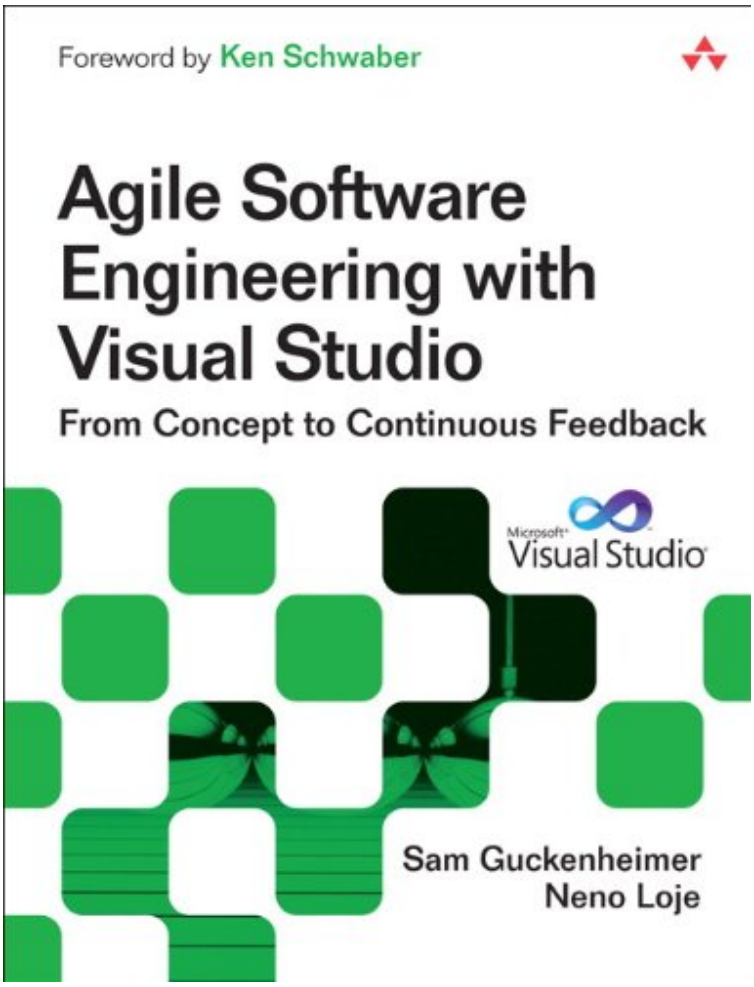
Agile Software Engineering with Visual Studio: From Concept to Continuous Feedback (Microsoft Windows Development Series)

Von Sam Guckenheimer, Neno Loje
*ebooks | Download PDF | *ePub | DOC | audiobook*



Produktinformation -Verkaufsrang: #787759 in eBooksVerffentlicht am: 2011-09-15Erscheinungsdatum: 2011-09-15File Name: B005N8EX1G | File size: 50.Mb

Von Sam Guckenheimer, Neno Loje : Agile Software Engineering with Visual Studio: From Concept to Continuous Feedback (Microsoft Windows Development Series) before purchasing it in order to gage whether or not it would be worth my time, and all praised Agile Software Engineering with Visual Studio: From Concept to Continuous Feedback (Microsoft Windows Development Series):

KundenrezensionenHilfreichste Kundenrezensionen1 von 1 Kunden fanden die folgende Rezension hilfreich. The definitive guide on agile development with Visual Studio 2010Von Ognjen BajicAgile software development methodologies have been proven on projects large and small. Nowadays methodology of choice is Scrum. It empowers multidisciplinary teams to successfully implement complex software and ensures the continuous flow of value through

the development process.This book will teach you how to successfully implement Scrum using integrated set of tools from Microsoft Visual Studio 2010 (VS) and the Team Foundation Server 2010 (TFS) and create automated process maximizing flow of value. In addition to the usual sprint and daily cycles, TFS based implementation of such process also exhibits micro cycles like check in and test. Ensuring the flow by making handoffs between team members as efficient as possible, by automating quality enforcing steps i.e. dones and gathering metrics without overhead at every cycle are cornerstones of this efficient process.The book goes beyond teaching you how to apply Scrum using VS and TFS. Reinforcing the flow of value by introducing removal of waste (bug debt, partially implemented features, unfinished code etc.) impeding the flow and transparency pinpointing the weak spots in the process, further ensure success of the development project. Guckenheimer and Loje teach how to identify different types of waste and deal with them. They do a great job explaining how to read different reports and analyze dashboards to gain real-time insight in progress, quality and other aspects of your project.VS and TFS aim at empowering the whole team.Architects can analyze legacy code or continuously validate the current architecture with every daily build using layer diagram.Developers will learn how to write clean code from the beginning and detect errors early. Different built in tools like check in policies or gated check-in help with that. Developers write or generate tests, check how effective they are and efficiently use them by executing only the tests impacted by a recent change from the set of all tests.Testers use simple but efficient tools to find bugs and fill rich bug reports (backed by video recording, debugger level Intellitrace information, test steps etc.). Such bugs are easily reproducible and can be quickly analyzed and resolved.Automated tests executed in virtualized test environments as a part of the build process with automated deployment are very powerful means to fight regressions.In the third part of the book, author shares valuable experiences from Microsoft, where the team that produces TFS and VS struggled with quality and schedule. By introducing many of the same techniques described in the book, in the last several years, they regained control. The last chapter shows glimpses of the future tools from the VS vNext.The authors of this book succeeded in two conflicting tasks: to offer clear, high level overview of modern agile software engineering practices on one side and on the other to dig deep enough in all the tools available in the Visual Studio 2010 and the TFS to show how they support these practices. Everything in an easy to read, moderately sized book.I highly recommend this book to anyone involved with the software development, irrespective of the role they play. You will learn the proven practices and the toolset as well as the rationale for the prescribed development process.1 von 1 Kunden fanden die folgende Rezension hilfreich. Praxisbuch fr jeden agilen Softwareentwickler im Microsoft .NET UmfeldVon Andreas RichterDie etwas ber 300 Seiten des Buches sind gefllt mit jeder Menge Informationen rund um den agilen Softwareprozess. Das Buch deckt einmal den kompletten agilen Prozess, beginnend mit den Grundlagen agiler Methoden, den Scrum-Artefakten, wie diese mit den .NET Tools Visual Studio, Team Foundation Server und Test Manager im Team umgesetzt werden knnen, bis hin zum Erfassen von Kundenfeedback ab. Was die beiden Autoren, bedingt durch den begrenzten Umfang des Buches, nicht aufnehmen konnten, haben sie konsequent mit weiterfhrenden Links versehen. Damit bleibt der Leser bei offenen Fragen nicht auf der Strecke sondern findet recht leicht detailliertere Informationen.Ein aus meiner Sicht idealer Begleiter fr .NET Entwicklungsteams, die in die Welt der agilen Softwareentwicklung eintauchen wollen oder bereits mitten drin sind. Das Buch sollte jeder gelesen haben, der Scrum, Kanban oder andere agile Methoden in den Weiten der .NET Entwicklung einsetzen mchte.1 von 1 Kunden fanden die folgende Rezension hilfreich. Real Life Meets TheoryVon Damir TomicicI really enjoyed reading this book and as it seems it is going to become one of my all time favorite books about software development with Visual Studio. It is written for a practicing audience and it demonstrates how the well-known practices and principles can help produce a high quality software product in the real life. Summed up - this is a must-read book for all Microsoft .NET developers.

KurzbeschreibungUsing agile methods and the tools of Visual Studio 2010, development teams can deliver higher-value software faster, systematically eliminate waste, and increase transparency throughout the entire development lifecycle. Now, Microsoft Visual Studio product owner Sam Guckenheimer and leading Visual Studio implementation consultant Neno Loje show how to make the most of Microsofts new Visual Studio 2010 Application Lifecycle Management (ALM) tools in your environment. This book is the definitive guide to the application of agile development with Scrum and modern software engineering practices using Visual Studio 2010. Youll learn how to use Visual Studio 2010 to empower and engage multidisciplinary, self-managing teams and provide the transparency they need to maximize productivity. Along the way, Guckenheimer and Loje help you overcome every major impediment that leads to stakeholder dissatisfactionfrom mismatched schedules to poor quality, blocked builds to irreproducible bugs, and technology silos to geographic silos. Coverage includesAccelerating the flow of value to customers in any software project, no matter how large or complexEmpowering high-performance software teams and removing overhead in software deliveryAutomating burndowns and using dashboards to gain a real-time, multidimensional view of quality and progressUsing Visual Studio 2010 to reduce or eliminate no repro bugsAutomating deployment and virtualizing test labs to make continuous builds deployableUsing Test Impact Analysis to quickly choose the right tests

based on recent code changesWorking effectively with sources, branches, and backlogs across distributed teamsSharing code, build automation, test, project and other data across .NET and Java teamsUncovering hidden architectural patterns in legacy software, so you can refactor changes more confidentlyScaling Scrum to large, distributed organizations Whatever your discipline, this book will help you use Visual Studio 2010 to focus on what really matters: building software that delivers exceptional value sooner and keeps customers happy far into the future. KurzbeschreibungUsing agile methods and the tools of Visual Studio 2010, development teams can deliver higher-value software faster, systematically eliminate waste, and increase transparency throughout the entire development lifecycle. Now, Microsoft Visual Studio product owner Sam Guckenheimer and leading Visual Studio implementation consultant Neno Loje show how to make the most of Microsofts new Visual Studio 2010 Application Lifecycle Management (ALM) tools in your environment. This book is the definitive guide to the application of agile development with Scrum and modern software engineering practices using Visual Studio 2010. Youll learn how to use Visual Studio 2010 to empower and engage multidisciplinary, self-managing teams and provide the transparency they need to maximize productivity. Along the way, Guckenheimer and Loje help you overcome every major impediment that leads to stakeholder dissatisfactionfrom mismatched schedules to poor quality, blocked builds to irreproducible bugs, and technology silos to geographic silos. Coverage includesAccelerating the flow of value to customers in any software project, no matter how large or complexEmpowering high-performance software teams and removing overhead in software deliveryAutomating burndowns and using dashboards to gain a real-time, multidimensional view of quality and progressUsing Visual Studio 2010 to reduce or eliminate no repro bugsAutomating deployment and virtualizing test labs to make continuous builds deployableUsing Test Impact Analysis to quickly choose the right tests based on recent code changesWorking effectively with sources, branches, and backlogs across distributed teamsSharing code, build automation, test, project and other data across .NET and Java teamsUncovering hidden architectural patterns in legacy software, so you can refactor changes more confidentlyScaling Scrum to large, distributed organizations Whatever your discipline, this book will help you use Visual Studio 2010 to focus on what really matters: building software that delivers exceptional value sooner and keeps customers happy far into the future. ber den Autor und weitere MitwirkendeSam Guckenheimer When I wrote the predecessor of this book, I had been at Microsoft less than three years. I described my history like this:I joined Microsoft in 2003 to work on Visual Studio Team System (VSTS), the new product line that was just released at the end of 2005. As the group product planner, I have played chief customer advocate, a role that I have loved. I have been in the IT industry for twenty-some years, spending most of my career as a tester, project manager, analyst, and developer. As a tester, I've always understood the theoretical value of advanced developer practices, such as unit testing, code coverage, static analysis, and memory and performance profiling. At the same time, I never understood how anyone had the patience to learn the obscure tools that you needed to follow the right practices. As a project manager, I was always troubled that the only decent data we could get was about bugs. Driving a project from bug data alone is like driving a car with your eyes closed and only turning the wheel when you hit something. You really want to see the right indicators that you are on course, not just feel the bumps when you stray off it. Here, too, I always understood the value of metrics, such as code coverage and project velocity, but I never understood how anyone could realistically collect all that stuff. As an analyst, I fell in love with modeling. I think visually, and I found graphical models compelling ways to document and communicate. But the models always got out of date as soon as it came time to implement anything. And the models just didn't handle the key concerns of developers, testers, and operations. In all these cases, I was frustrated by how hard it was to connect the dots for the whole team. I loved the idea in Scrum (one of the Agile processes) of a "single product backlog"--one place where you could see all the work--but the tools people could actually use would fragment the work every which way. What do these requirements have to do with those tasks, and the model elements here, and the tests over there? And where's the source code in that mix? From a historical perspective, I think IT turned the corner when it stopped trying to automate manual processes and instead asked the question, "With automation, how can we reengineer our core business processes?" That's when IT started to deliver real business value. They say the cobbler's children go shoeless. That's true for IT, too. While we've been busy automating other business processes, we've largely neglected our own. Nearly all tools targeted for IT professionals and teams seem to still be automating the old manual processes. Those processes required high overhead before automation, and with automation, they still have high overhead. How many times have you gone to a 1-hour project meeting where the first 90 minutes were an argument about whose numbers were right? Now, with Visual Studio, we are seriously asking, "With automation, how can we reengineer our core IT processes? How can we remove the overhead from following good process? How can we make all these different roles individually more productive while integrating them as a high performance team?" Obviously, that's all still true. Neno Loje I started my career as a software developer--first as a hobby, later as profession. At the beginning of high school, I fell in love with writing software because it enabled me to create something useful by transforming an idea into something of actual value for someone else. Later, I learned that this was generating customer value. However, the impact and value were limited by the fact that I was just a single developer working in a small company, so I decided to focus on helping and teaching other developers. I started by delivering pure technical training, but the topics soon expanded to include process and people, because I realized that just introducing a new tool or a technology

by itself does not necessarily make teams more successful. During the past six years as an independent ALM consultant and TFS specialist, I have helped many companies set up a team environment and software development process with VS. It has been fascinating to watch how removing unnecessary, manual activities makes developers and entire projects more productive. Every team is different and has its own problems. I've been surprised to see how many ways exist (both in process and tools) to achieve the same goal: deliver customer value faster though great software. When teams look back at how they worked before, without VS, they often ask themselves how they could have survived without the tools they use now. However, what had changed from the past were not only the tools, but also the way they work as a team. Application Lifecycle Management and practices from the Agile Consensus help your team to focus on the important things. VS and TFS are a pragmatic approach to implement ALM (even for small, nondistributed teams). If you're still not convinced, I urge you to try it out and judge for yourself.