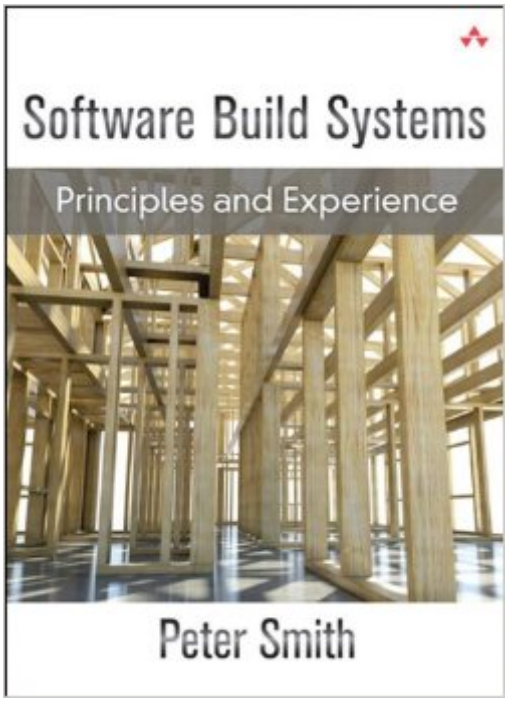
Software Build Systems: Principles And Experience



Synopsis

â œThis book represents a thorough and extensive treatment of the software build process including the choices, benefits, and challenges of a well designed build process.Â I recommend it not only to all software build engineers but to all software developers since a well designed build process is key to an effective software development process.â • â ”Kevin Bodie, Director Software Development, Pitney Bowes Inc. Â â œAn excellent and detailed explanation of build systems, an important but often overlooked part of software development projects. The discussion of productivity as related to build systems is, alone, well worth the time spent reading this book.â • â ”John M. Pantone, Objectech Corporation, VP, IT Educator and Course Developer Â â œPeter Smith provides an interesting and accessible look into the world of software build systems, distilling years of experience and covering virtually every type of tool in the build engineerâ ™s toolbox. Well organized, well written, and very thorough; I would recommend this book to anyone with a build system under their responsibility.â • â ”Jeff Overbey, Project Co-Lead, Photran Â â œSoftware Build Systems teaches how to think about building software. It surveys the tools and techniques for building software products and the ways things go wrong. This book will appeal to those new to build systems as well as experienced build system engineers.â • â ”Monte Davidoff, Software Development Consultant, Alluvial Software, Inc. Â Inadequate build systems can dramatically impact developer productivity. Bad dependencies, false compile errors, failed software images, slow compilation, and time-wasting manual processes are just some of the byproducts of a subpar build system. In Software Build Systems, software productivity expert Peter Smith shows you how to implement build systems that overcome all these problems, so you can deliver reliable software more rapidly, at lower cost. Â Smith explains the core principles underlying highly efficient build systems, surveying both system features and usage scenarios. Next, he encapsulates years of experience in creating and maintaining diverse build systemsâ “helping you make well-informed choices about tools and practices, and avoid common traps and pitfalls. Throughout, he shares a wide range of practical examples and lessons from multiple environments, including Java, C++, C, and C#. Coverage includes Â â ¢ Mastering build system concepts, including source trees, build tools, and compilation tools â ¢ Comparing five leading build tools: GNU Make, Ant, SCons, CMake, and the Eclipse IDEâ ™s integrated build features â ¢ Ensuring accurate dependency checking and efficient incremental compilation â ¢ Using metadata to assist debugging, profiling, and source code documentation â ¢ Packaging software for installation on your target machine â ¢ Best practices for managing complex version-control systems, build machines, and compilation tools Â If youâ ™re a developer, this book will illuminate the issues involved in building and maintaining the build system

thatâ ™s best for your team. If youâ ™re a manager, youâ ™ll discover how to evaluate your teamâ ™s build system and improve its effectiveness. And if youâ ™re a build â œguru,â • youâ ™ll learn how to optimize the performance and scalability of your build system, no matter how demanding your requirements are.

Book Information

File Size: 9435 KB

Print Length: 624 pages

Simultaneous Device Usage: Up to 5 simultaneous devices, per publisher limits

Publisher: Addison-Wesley Professional; 1 edition (March 11, 2011)

Publication Date: March 11, 2011

Sold by:Â Digital Services LLC

Language: English

ASIN: B004SH6XIE

Text-to-Speech: Enabled

X-Ray: Not Enabled

Word Wise: Not Enabled

Lending: Not Enabled

Enhanced Typesetting: Enabled

Best Sellers Rank: #404,511 Paid in Kindle Store (See Top 100 Paid in Kindle Store) #41 inÂ Books > Computers & Technology > Programming > Languages & Tools > Compiler Design #102 inÂ Books > Computers & Technology > Programming > Languages & Tools > Compilers #440 inÂ Kindle Store > Kindle eBooks > Computers & Technology > Programming > Software Design > Software Development

Customer Reviews

Advertising "Principles and Experience" with software build tools, theauthor demonstrates a surprisingly sophomoric depth ofunderstanding. Most of the book is presented at a very high level withlimited practical presentation.Chapter 08 discusses the SCons tool with a section on debugging badbuilds. The --debug=presub command-line switch is introduced andoutput presented. A particular environment variable is mentioned andthen some portion of the list of all environment variable definitionsin SCons is printed. The actual environment variable presented isnever found in the list, the result is never plugged into the templatecommand-line, and the "problem" is never debugged. The level ofunderstanding seems very much that of someone who has run the

on-linetutorial.The feeling of reiterated tutorial continues in Chapter 10 onEclipse. I can figure out the panes of the workspace or else I canfire up the Guided Tour that ships with the product. There is no needto spend several pages on helping me figure out where source code isdisplayed.Chapter 06 focuses on GNU Make. The idea of a .flags file is mentionedas a way to capture dependencies on command-line flags. Where is anexample? Is it better practice to have a .flags file per directory orper source file? How do .flag files interact with the number of filereads done by a build system (the topic of Chapter 19 when talkingabout speeding up a build)?There are a handful of editing errors and another, more annoying,handful of technical errors. There is a discussion of running gdb(why, in a book on building, do I get a tutorial on gdb?

A Gap-Filling Book for Software Developers TooBuild system are often viewed as merely a script to compile the code. However there are much more to it. Read this book you'll find out. As an editorial review says, this book is also for developers in addition to build engineers.As the book starts with, a survey showed that it is common developers losing 20~30% productivity due to build related problems. My personal estimate, at one of my previous work we spent more than 30% of time dealing with build breakages. Sometimes the problems are caused by the build system itself, some other times they are due to misuse of the system. It's worth for developers to learn the build system thus to avoid messing up with the build system.For large software it is very difficult to figure out the dependencies among the components. To my view, if it is a C program, the only place the complete dependency information can be found is in the build system. Figuring out dependencies helps developers to better design/code/debug their software. To do so you'll have to know about the build system, or you may even need to fix the build system so you can get the information dumped for you.I use code generation in my coding practice so that I can manage more logic in code and avoid tedious repetitive work [see Martin Fowler's DSL book Domain-Specific Languages (Addison-Wesley Signature Series (Fowler)) for more on this topic]. I also generate tests from the source code. To generate code the triggers have to be incorporated into the build system. Had I read this book I would have done my work more effectively.

This book is a decent introduction to the world of build systems. Overall I expected more depth from it, and felt I barely got my money's worth, but it could be perfect for someone with less experience with complex build systems who wants to learn more about the field or got tasked with choosing and implementing a build system for a project.It starts with a high-level discussion of concepts that are common across build systems. Then it goes into an overview of five common build systems - make,

scons, cmake, Ant and the GUI-based Eclipse. Generally I felt this is the weakest part of the book as it does not go into depth with any of those tools, but it is useful if one wants to compare how several systems handle a common, simple scenario, and the praise/criticism/evaluation section towards the end of each tool’s description is a nice summary. For a detailed manual of one of the covered build tools, tool documentation and other in-depth books on separate tools are a far better choice.Part III, "Advanced topics", is a misnomer - it should be called "Beyond the basics". Once again the discussion becomes more abstract and covers dependencies and related issues, "metadata" (any products that are not the final executables - documentation, tests, etc.), managing tools, and towards the end two topics that are the most important - how to improve build speed and manage build size. I would not call any topic discussed to actually be "advanced" - that would be, for example, an in-depth analysis of an existing complex make-based build framework, or a careful dissection, with reasoning and trade-offs, of a big build system of an existing open source project, none of which are to be seen in this book.

Decks For Ponds Step by Step: Step by step guide with images and plans to build a floating dock pier and a farm pond deck. BUSINESS:Business Marketing, Innovative Process How To Startup, Grow And Build Your New Business As Beginner, Step By Step Online Guide How To Effective ... Grow And Build Business As Beginner) Agile Software Development, Principles, Patterns, and Practices Build APIs You Won't Hate: Everyone and their dog wants an API, so you should probably learn how to build them How to Build a Computer: Learn How to Build Your Own Computer From Scratch. The Parts, Connecting Everything Together, Installation and more (PC, Windows, Gaming System, Media System, Linux) Inside the Microsoft Build Engine: Using MSBuild and Team Foundation Build (Developer Reference)