O'REILLY®



User Story Mapping

DISCOVER THE WHOLE STORY,
BUILD THE RIGHT PRODUCT

Jeff Patton
with Peter Economy

Forewords by Martin Fowler,
Alan Cooper, and Marty Cagan

Documents
Document Sharing Hub

Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

O'REILLY®

User Story Mapping

User story mapping is a valuable tool for software development, once you understand why and how to use it. This insightful book examines how this often misunderstood technique can help your team stay focused on users and their needs without getting lost in the enthusiasm for individual product features.

Author Jeff Patton shows you how changeable story maps enable your team to hold better conversations about the project throughout the development process. Your team will learn to come away with a shared understanding of what you're attempting to build and why.

- Get a high-level view of story mapping, with an exercise to learn key concepts quickly
- Understand how stories really work, and how they come to life in Agile and Lean projects
- Dive into a story's lifecycle, starting with opportunities and moving deeper into discovery
- Prepare your stories, pay attention while they're built, and learn from those you convert to working software

"I have met only a few Agile experts whom I consider qualified to actually help a serious product team raise its game to the level its company needs and deserves. Jeff Patton is one of them."

—**Marty Cagan**
Partner, Silicon Valley Product Group

Jeff Patton is an independent consultant, agile process coach, product design process coach, and instructor with more than 15 years of experience designing and building software products. He's been focused on agile approaches since working on an early extreme programming team in 2000.

SOFTWARE DESIGN

US $34.99 **CAN $36.99**
ISBN: 978-1-491-90490-9





Twitter: @oreillymedia
facebook.com/oreilly

Documents
Document Sharing Hub

Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

User Story Mapping
by Jeff Patton

Copyright © 2014 Jeff Patton. All rights reserved.

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://safaribooksonline.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Editors: Mary Treseler and Amy Jollymore
Production Editor: Kara Ebrahim
Copyeditor: Rachel Monaghan
Proofreader: Elise Morrison
Indexer: Ellen Troutman
Cover Designer: Ellie Volckhausen
Interior Designer: David Futato
Illustrator: Rebecca Demarest

September 2014: First Edition

Revision History for the First Edition:

2014-09-05: First release

See *http://oreilly.com/catalog/errata.csp?isbn=9781491904909* for release details.

Nutshell Handbook, the Nutshell Handbook logo, and the O'Reilly logo are registered trademarks of O'Reilly Media, Inc. *User Story Mapping*, the image of a lilac-breasted roller, and related trade dress are trademarks of O'Reilly Media, Inc.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly Media, Inc. was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

ISBN: 978-1-491-90490-9

[LSI]

Documents
Document Sharing Hub

Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

Table of Contents

Foreword by Martin Fowler xi
Foreword by Alan Cooper xiii
Foreword by Marty Cagan xvii
Preface xxi
Read This First xxix

1. The Big Picture 1
The "A" Word 1
Telling Stories, Not Writing Stories 3
Telling the Whole Story 3
Gary and the Tragedy of the Flat Backlog 5
Talk and Doc 6
Frame Your Idea 8
Describe Your Customers and Users 9
Tell Your Users' Stories 10
Explore Details and Options 14

2. Plan to Build Less 21
Mapping Helps Big Groups Build Shared Understanding 22
Mapping Helps You Spot Holes in Your Story 25
There's Always Too Much 26
Slice Out a Minimum Viable Product Release 27
Slice Out a Release Roadmap 28
Don't Prioritize Features—Prioritize Outcomes 29
This Is Magic—Really, It Is 30
Why We Argue So Much About MVP 32
The New MVP Isn't a Product at All! 34

Documents
Document Sharing Hub

Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

3. Plan to Learn Faster. 37
Start by Discussing Your Opportunity 38
Validate the Problem 39
Prototype to Learn 40
Watch Out for What People Say They Want 41
Build to Learn 41
Iterate Until Viable 44
How to Do It the Wrong Way 44
Validated Learning 46
Really Minimize Your Experiments 48
Let's Recap 48

4. Plan to Finish on Time. 51
Tell It to the Team 52
The Secret to Good Estimation 53
Plan to Build Piece by Piece 54
Don't Release Each Slice 56
The Other Secret to Good Estimation 56
Manage Your Budget 57
What Would da Vinci Do? 59
Iterative AND Incremental 62
Opening-, Mid-, and Endgame Strategy 63
Slice Out Your Development Strategy in a Map 64
It's All About Risk 64
Now What? 65

5. You Already Know How. 67
1. Write Out Your Story a Step at a Time 67
Tasks Are What We Do 68
My Tasks Are Different Than Yours 69
I'm Just More Detail-Oriented 70
2. Organize Your Story 71
Fill in Missing Details 72
3. Explore Alternative Stories 72
Keep the Flow 74
4. Distill Your Map to Make a Backbone 75
5. Slice Out Tasks That Help You Reach a Specific Outcome 76
That's It! You've Learned All the Important Concepts 77
Do Try This at Home, or at Work 78
It's a Now Map, Not a Later Map 79
Try This for Real 81

Documents
Document Sharing Hub

Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

With Software It's Harder 82
The Map Is Just the Beginning 84

6. The Real Story About Stories. 89
Kent's Disruptively Simple Idea 89
Simple Isn't Easy 91
Ron Jeffries and the 3 Cs 92
1. Card 93
2. Conversation 93
3. Confirmation 94
Words and Pictures 95
That's It 96

7. Telling Better Stories. 97
Connextra's Cool Template 97
Template Zombies and the Snowplow 102
A Checklist of What to Really Talk About 104
Create Vacation Photos 107
It's a Lot to Worry About 108

8. It's Not All on the Card. 109
Different People, Different Conversations 109
We're Gonna Need a Bigger Card 110
Radiators and Ice Boxes 113
That's Not What That Tool Is For 116
Building Shared Understanding 116
Remembering 118
Tracking 119

9. The Card Is Just the Beginning. 121
Construct with a Clear Picture in Your Head 122
Build an Oral Tradition of Storytelling 123
Inspect the Results of Your Work 124
It's Not for You 126
Build to Learn 127
It's Not Always Software 128
Plan to Learn, and Learn to Plan 129

10. Bake Stories Like Cake. 131
Create a Recipe 132
Breaking Down a Big Cake 133

Documents
Document Sharing Hub
Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

11. Rock Breaking. 137
Size Always Matters 137
Stories Are Like Rocks 139
Epics Are Big Rocks Sometimes Used to Hit People 140
Themes Organize Groups of Stories 142
Forget Those Terms and Focus on Storytelling 142
Start with Opportunities 143
Discover a Minimum Viable Solution 144
Dive into the Details of Each Story During Delivery 146
Keep Talking as You Build 148
Evaluate Each Piece 149
Evaluate with Users and Customers 150
Evaluate with Business Stakeholders 152
Release and Keep Evaluating 153

12. Rock Breakers. 155
Valuable-Usable-Feasible 156
A Discovery Team Needs Lots of Others to Succeed 158
The Three Amigos 159
Product Owner as Producer 163
This Is Complicated 164

13. Start with Opportunities. 167
Have Conversations About Opportunities 167
Dig Deeper, Trash It, or Think About It 168
Opportunity Shouldn't Be a Euphemism 173
Story Mapping and Opportunities 173
Be Picky 179

14. Using Discovery to Build Shared Understanding. 181
Discovery Isn't About Building Software 181
Four Essential Steps to Discovery 182
1. Frame the Idea 183
2. Understand Customers and Users 183
3. Envision Your Solution 186
4. Minimize and Plan 196
Discovery Activities, Discussions, and Artifacts 199
Discovery Is for Building Shared Understanding 200

15. Using Discovery for Validated Learning. 201
We're Wrong Most of the Time 201

Documents
Document Sharing Hub

Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

The Bad Old Days 203
Empathize, Focus, Ideate, Prototype, Test 204
How to Mess Up a Good Thing 208
Short Validated Learning Loops 209
How Lean Startup Thinking Changes Product Design 210
Start by Guessing 211
Name Your Risky Assumptions 212
Design and Build a Small Test 212
Measure by Running Your Test with Customers and Users 214
Rethink Your Solution and Your Assumptions 215
Stories and Story Maps? 215

16. Refine, Define, and Build. 217
Cards, Conversation, More Cards, More Conversations… 217
Cutting and Polishing 218
Workshopping Stories 218
Sprint or Iteration Planning? 222
Crowds Don't Collaborate 225
Split and Thin 227
Use Your Story Map During Delivery 232
Use a Map to Visualize Progress 233
Use Simple Maps During Story Workshops 234

17. Stories Are Actually Like Asteroids. 239
Reassembling Broken Rocks 241
Don't Overdo the Mapping 243
Don't Sweat the Small Stuff 244

18. Learn from Everything You Build. 247
Review as a Team 247
Review with Others in Your Organization 251
Enough 253
Learn from Users 254
Learn from Release to Users 255
Outcomes on a Schedule 255
Use a Map to Evaluate Release Readiness 256

The End, or Is It?. 259

Acknowledgments. 261

Documents
Document Sharing Hub
Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

Foreword by Martin Fowler

One of the beneficial consequences of the rise of Agile software development is the notion of splitting up large sets of requirements into smaller chunks. These chunks—stories—enable much more visibility into the progress of a development project. When a product is built story-by-story, with each story's implementation fully integrated into the software product, everyone can see the product grow. By using stories that make sense to users, developers can steer the project by determining which stories to build next. This greater visibility helps encourage greater participation from users—no longer do they have to wait a year or more to see what the development team's been up to.

But this chunking has some negative consequences. One of these is that it's easy to lose the big picture of what a software system should do. You can end up with a jumble of pieces that don't fit into a coherent whole. Or you can end up building a system that isn't really helpful to the users, because you've missed the essence of what's needed by getting lost in the details.

Story mapping is a technique that provides the big picture that a pile of stories so often misses.

That's it, really—the description of this book in a single sentence. And that sentence carries with it the promise of a lot of value. A big picture helps communicate effectively with users, it helps everyone involved avoid building unnecessary features, and it provides an orientation for a coherent user experience. When I talk to my colleagues at ThoughtWorks about what they do to develop their stories, story mapping regularly comes up as a core technique. Often they've learned that technique from workshops run by Jeff, because he's the one who developed the technique and can best communicate it. This book

Documents
Document Sharing Hub

Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

allows more people to understand this technique directly from its source.

But this isn't just a book for people who have something like "business analyst" on their business card or online profile. Perhaps the biggest disappointment for me in the decade of the adoption of Agile methods is the way that many programmers see stories as a one-way communication from analysts to them. Right from the beginning, stories were supposed to spark *conversations*. If you really want to come up with effective software to support an activity, then you need to look to those who build software as a vital source of ideas for its capabilities, because it's programmers who know best what software can do. Programmers need to understand what their users are trying to achieve and should collaborate in building the stories that capture those users' needs. A programmer who understands story mapping can better see the broader user context and can participate in framing the software—leading to a better job.

When Kent Beck (who originated the notion of a "story") developed his ideas on software development, he called out communication as a key value of effective teams. Stories are the building blocks of communication between developers and those who use their work. Story maps organize and structure these building blocks, and thus enhance this communication process—which is the most critical part of software development itself.

—Martin Fowler
June 18, 2014

Documents
Document Sharing Hub

Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

Foreword by Alan Cooper

In Mary Shelley's famous science-fiction novel, *Frankenstein*, the mad Doctor Frankenstein builds a creature from disparate pieces of dead humans and brings the creature to life with the then-new technology of electricity. Of course, we know that this is not actually possible. You cannot create life by sewing together random body parts.

Yet this is what software developers attempt to do all the time. They add good features to software, one at a time, and then wonder why few users love their product. The heart of the conundrum is that developers are using their construction method as a design tool, but the two are not interchangeable.

It's entirely reasonable that programmers *build* software one feature at a time. That's a perfectly good strategy, proven over the years. What has also been proven over the years is that, when used as a method for designing the behavior and scope of a digital product, one-feature-at-a-time yields a Frankenstein monster of a program.

While they are intimately related, the practice of designing software behavior and the practice of building that software are distinctly different, and are typically performed by different people with different skill sets. The many hours that interaction designers spend observing users and mapping behavior patterns would drive most programmers batty. Conversely, the hours of sweating over algorithms are too solitary for most designers.

But when the two strains of practice—design and development—collaborate, the work becomes electric and has the potential to create a living, breathing product. Teamwork breathes life into the monster and makes people love it.

Documents
Document Sharing Hub

Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

While the idea of collaboration is neither new nor particularly insightful, it is actually very difficult to do effectively. The way that developers work—their pace, language, and rhythm—is quite different from that of interaction designers.

Practitioners in each of the two fields are strong, capable, and internally well disciplined, yet they share a single, common weakness. It is really hard to express a design problem in programming terms, and it is equally hard to express a development problem in design terms. The two sister disciplines lack a common tongue. And that junction between the two disciplines is precisely where Jeff Patton lives.

Jeff's method of story mapping makes sense to developers, and it makes equal sense to designers. Story mapping is the Rosetta Stone for our digital age.

Despite protestations to the contrary, Agile development is not a very useful design tool. It is a way of thinking about development that is design-friendly, which is a very good thing, but by itself it won't get you to a product that users love. On the other hand, so many times we have seen good designs, well documented, given to developers—Agile or not—who manage to kill the essence of the design in the process of implementation.

Patton's story mapping approach is the bridge over this chasm. Interaction design is all about finding the user's truth and telling it as a narrative. Software development is all about breaking those narratives into tiny, functional chunks and implementing and integrating them. It's so ridiculously easy for the essence of the narrative to slip away during this complex process. Yes, the functions are implemented, but the patient dies on the operating room table.

By mapping out the user's stories, the design retains its narrative structure yet can still be deconstructed for effective implementation. The designer's story, which is a formalized version of the user's story, remains intact throughout the development.

The conventional corporate world has proven that it is nearly impossible for a team of two or three hundred people to build a product that people love. Meanwhile the startup community has proven that a team of four or five people *can* build small products that people love, but even these little products eventually grow big and lose their spark. The challenge we face is creating big software that people love. Big software

Documents
Document Sharing Hub

Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

serves large audiences doing complex, commercially viable jobs. It's ridiculously hard to make such software fun to use and easy to learn.

The only way we are going to build big software that is not a Frankenstein monster is by learning how to integrate the disciplines of software design and development. Nobody knows how to do that better than Jeff Patton.

—Alan Cooper
June 17, 2014

Documents
Document Sharing Hub
Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

Documents
Document Sharing Hub

Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

Foreword by Marty Cagan

I've had the extremely good fortune to be able to work with many of the very best technology product teams in the world. People creating the products you use and love every day. Teams that are literally changing the world.

I've also been brought in to try to help companies that are not doing so well. Startups racing to get some traction before the money runs out. Larger companies struggling to replicate their early innovation. Teams failing to continuously add value to their business. Leaders frustrated with how long it takes to go from idea to reality. Engineers exasperated with their product owners.

What I've learned is that there is a profound difference between how the very best product companies create technology products, and the rest. And I don't mean minor differences. I mean everything from how leaders behave to the level of empowerment of teams; to the way teams work together; to how the organization thinks about funding, staffing, and producing products; to the culture; to how product, design, and engineering collaborate to discover effective solutions for their customers.

This book is titled *User Story Mapping*, but you'll soon see it is about much more than this powerful yet simple technique. This book gets to the heart about how teams collaborate, communicate, and ultimately come up with good stuff to build.

Many of you have never had a chance to see up close how a strong product team operates. All you may know is what you've seen at your company or where you've worked before. So what I'd like to do here

Documents
Document Sharing Hub

Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

is to try to give you a flavor of just how different the best teams are from the rest.

With a grateful nod to Ben Horowitz's *Good Product Manager, Bad Product Manager*, here's a glimpse into some of the important differences between strong product teams and weak teams:

> Good teams have a compelling product vision that they pursue with a missionary-like passion. Bad teams are mercenaries.
>
> Good teams get their inspiration and product ideas from their scorecard KPIs, from observing customers struggle, from analyzing the data customers generate from using their product, and from constantly seeking to apply new technology to solve real problems. Bad teams gather requirements from sales and customers.
>
> Good teams understand who their key stakeholders are, they understand the constraints that these stakeholders operate in, and they are committed to inventing solutions that not only work for users and customers, but also work within the constraints of the business. Bad teams gather requirements from stakeholders.
>
> Good teams are skilled in the many techniques to rapidly try out product ideas to determine which ones are truly worth building. Bad teams hold meetings to generate prioritized roadmaps.
>
> Good teams love to have brainstorming discussions with smart thought leaders from across the company. Bad teams get offended when someone outside their team dares to suggest they do something.
>
> Good teams have product, design, and engineering sit side-by-side, and embrace the give and take between the functionality, the user experience, and the enabling technology. Bad teams sit in their respective functional areas, and ask that others make requests for their services in the form of documents and scheduling meetings.
>
> Good teams are constantly trying out new ideas in order to innovate, but doing so in ways that protect the revenue and the brand. Bad teams are still waiting for permission to run a test.
>
> Good teams insist they have the skill sets necessary to create winning products, such as strong interaction design. Bad teams don't even know what interaction designers are.
>
> Good teams ensure that their engineers have time to try out the discovery prototypes every day so that they can contribute their thoughts on how to make the product better. Bad teams show the prototypes to the engineers during sprint planning so they can estimate.
>
> Good teams engage directly with end users and customers every week to better understand their customers, and to see the customer's response to their latest ideas. Bad teams think they are the customer.

Documents
Document Sharing Hub
Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

> Good teams know that many of their favorite ideas won't end up working for customers, and even the ones that could will need several iterations to get to the point where they provide the desired outcome. Bad teams just build what's on the roadmap and are satisfied with meeting dates and ensuring quality.
>
> Good teams understand the need for speed and how rapid iteration is the key to innovation, and they understand this speed comes from the right techniques and not forced labor. Bad teams complain they are slow because their colleagues are not working hard enough.
>
> Good teams make high-integrity commitments after they've evaluated the request and ensured they have a viable solution that will actually work for the customer and the business. Bad teams complain about being a sales-driven company.
>
> Good teams instrument their work so that they can immediately understand how their product is being used and make adjustments based on the data. Bad teams consider analytics and reporting a "nice to have."
>
> Good teams integrate and release continuously, knowing that a constant stream of smaller releases provides a much more stable solution for their customers. Bad teams test manually at the end of a painful integration phase and then release everything at once.
>
> Good teams obsess over their reference customers. Bad teams obsess over competitors.
>
> Good teams celebrate when they achieve a significant impact to the business KPIs. Bad teams celebrate when they finally release something.

I realize you might be wondering what all this has to do with story maps. I think you'll be surprised. And that's precisely why I am a fan of story maps.

I have met only a few Agile experts whom I consider qualified to actually help a serious product team raise its game to the level its company needs and deserves. Jeff Patton is one of them. I have observed him working hands on in the trenches with teams in the midst of product discovery. I introduce him into companies because he is effective. Teams love him because he is knowledgeable yet humble.

The days of product managers gathering up and documenting requirements, designers scrambling just to put some lipstick on the product, and engineers sheltered in the basement, coding, are long gone for the best teams. And it's time they are gone for your team, too.

—Marty Cagan
June 18, 2014

Documents
Document Sharing Hub
Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

Documents
Document Sharing Hub

Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

Preface

> Live in it, swim in it, laugh in it, love in it / Removes embarrassing stains from contour sheets, that's right / And it entertains visiting relatives, it turns a sandwich into a banquet.
>
> — Tom Waits, "Step Right Up"

This book was supposed to be a small thing…a pamphlet, really.

I set out to write about a simple practice I called *story mapping*. I, and lots of other folks, build simple maps to help us work together with others and to imagine the experience of using a product.

Story mapping keeps us focused on users and their experience, and the result is a better conversation, and ultimately a better product.



Building a map is dead simple. Working together with others, I'll tell the story of a product, writing each big step the users take in the story on sticky notes in a left-to-right flow. Then, we'll go back and talk

Documents
Document Sharing Hub
Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

about the details of each step, and write those details down on sticky notes and place them vertically under each step. The result is a simple grid-like structure that tells a story from left to right, and breaks it into details from top to bottom. It's fun and fast. And those details make a better backlog of stories for our Agile development projects.

How complicated could writing a book about this be?

But it turns out that even the simple things can be pretty sophisticated. And writing about why you would want to build a story map, what's going on when you build one, and all the different ways you can use one took me a lot of pages. There was more to this simple practice than I thought.

If you're using an Agile development process, you're likely filling backlogs with user stories. I'd assumed that since stories were such a common practice, it'd be a waste of time for me to write about them in this book. But I was wrong. In the decade and a half since stories were first described by Kent Beck, they're more popular—and more misunderstood and misused—than ever before. That makes me sad. And, what's more, it kills all the benefit we get from story mapping.

So, in this book, I would like to correct as many big misconceptions as I can about stories and the way they're used in Agile and Lean software development. That's why, in the words of Tom Waits, I've turned this "sandwich into a banquet."

Why Me?

I like making things. What motivates me is the joy I get from creating a piece of software and seeing people use it and benefit from it. I'm a reluctant methodologist. I found I needed to learn how process and practice work to get better at them. I'm only now learning after 20-plus years in software development how to teach what I've learned. And I know that what I teach is a moving target. What I understand changes every week. How best to explain it changes almost as fast. All that's kept me from writing a book for years.

But it's time.

Stories and story maps are such a good idea. They've benefited so many people. They've made their lives better, and the products they build better. But while some people's lives are getting better, there are more

Documents
Document Sharing Hub
Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

people struggling with stories than ever before. I want to help stop that.

This book is something I can make to help. And, if it improves the work lives of even a few, I'll celebrate.

This Book Is for You If You're Struggling with Stories

Because so many organizations have adopted Agile and Lean processes, and stories along with them, you may fall into one or more of the traps caused by misconceptions about stories. Traps like these:

- Because stories let you focus on building small things, it's easy to *lose sight of the big picture*. The result is often a "Franken-product" where it's clear to everyone using the product that it's assembled from mismatched parts.
- When you're building a product of any significant size, building one small thing after another leaves people *wondering when you'll ever be done, or what exactly you'll deliver*. If you're the builder, you wonder, too.
- Because stories are about conversations, *people use that idea to avoid writing anything down*. Then they forget what they talked about and agreed to in the conversations.
- Because good stories are supposed to have acceptance criteria, we focus on getting acceptance criteria written, but there's still not a common understanding of what needs to be built. As a consequence, *teams don't finish the work they plan on in the timeframe they planned to*.
- Because good stories are supposed to be written from a user's perspective, and there are lots of parts that users never see, team members argue that "*our product doesn't have users, so user stories won't work here.*"

If you've fallen into any of those traps, then I'll try to wipe away the misconceptions that lead to those traps in the first place. You'll learn how to think of the big picture, how to plan and estimate in the large (and in the small), and how to have productive conversations about what users are trying to accomplish, as well as what a good piece of software needs to do to help them.

Documents
Document Sharing Hub
Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

Who Should Read This Book?

You should, of course. Especially if you bought it. I, for one, think you've made a wise investment. If you're just borrowing it, you should order your own now, and return the one you've borrowed when the new one arrives at your door.

However, reading this book offers specific reasons and benefits for practitioners in specific roles:

- *Product managers and user experience (UX) practitioners in commercial product companies* should read this book to help them bridge the gap between thinking about whole products and user experience and thinking about tactical plans and backlog items. If you've been struggling to get from the vision you're imagining to the details your teams can build, story maps will help. If you've been struggling to help others imagine the experience of—and empathize with—the users of your product, story mapping will help. If you've been struggling to figure out how to incorporate good UX and product design practice, this book will help. If you've been working to incorporate Lean Startup–style experimentation in the way you work, this book will help.
- *Product owners, business analysts, and project managers in information technology (IT) organizations* should read this book to help them bridge the gap between their internal users, stakeholders, and developers. If you've been struggling to convince lots of stakeholders in your company to get on the same page, then story maps will help. If you've been struggling to help developers see the big picture, story maps will help.
- *Agile and Lean process coaches* with the goal of helping individuals and teams improve should read this book. And, as you do, think about the misconceptions people in your organization have about stories. Use the stories, simple exercises, and practices described in this book to help your teams improve.
- *Everyone else*. When using Agile processes, we often look to roles like product owners or business analysts to steer a lot of the work with stories, but effective use of stories requires that *everyone* get the basics. When people don't understand the basics, you hear complaints that "stories aren't well written" or that they're "too big," or that they "don't have enough detail." This book will help, but not in the way you think. You and everyone else will learn that

Documents
Document Sharing Hub
Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

stories aren't a way to write better requirements, but a way to organize and have better conversations. This book will help you understand what kinds of conversations you should be having to help you get the information you need when you need it.

I'm hoping you identify with one or more of the groups I just described. If you don't, give this book to someone who does.

If you do, let's get started.

A Few Conventions Used in This Book

I suspect this isn't the only book on software development you've ever read, so nothing should surprise you.

The Headings Inside Each Chapter Guide You Through the Subject

Use them to find your way or skip over stuff you're not interested in right now.

> *Key points look like this. Imagine me saying these a bit louder than all the other text.*

If you're skimming, read the key points. If you like them, or they're not dead obvious, read the text before and after them. That should make them clear.

> Sidebars are used to describe:
>
> - *Interesting but not critical concepts*. These should be fun distractions. At least I hope they are.
> - *Recipes for specific practices*. You should be able to use these recipes to help you get started with a specific practice.
> - *Stories and examples contributed by others*. You should get some good ideas from these that you could try in your organization.

The book is organized into specific sections. You could read it a section at a time, or use the sections to help you find ideas for a specific challenge you have right now.

Documents
Document Sharing Hub
Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

How This Book Is Organized

I bought a cool new color laser printer a while back. I opened the box, and sitting on top of the printer was a pamphlet with "Read This First" in big red letters on it. I wondered, "Should I *really* read this first?" because I usually don't do as I'm told. But I'm glad I did, because there were lots of plastic guards in various places inside the printer to keep it safe during shipping, and if I'd plugged it in before removing them, I might have damaged the printer.

This story might sound like a tangent, but it's not.

This book contains a "Read This First" chapter because there are two critical concepts and associated vocabulary that I'll use throughout the rest of the book. I'd like you to have those concepts in your head before you get started. If you start to story map before you understand them, I can't guarantee your safety.

Story Mapping from 10,000 Feet

Chapters 1–4 will give you a high-level view of story mapping. If you've been using stories for a while and played with a story map before, this section should give you enough to get going right away.

Chapter 5 gives you a nifty exercise to help you learn the key concepts used to create a great story map. Try it out with a group in your office, and everyone who participates will get it. And I promise you the maps they create for your products will come out better afterward.

Grokking User Stories

Chapters 6–12 tell the story behind stories, how they really work, and how to make good use of them in Agile and Lean projects. Inside story maps are lots of little stories you can use to drive day-to-day development. Even if you're an Agile veteran, I promise you'll learn something about stories you didn't already know. And, if you're new to stories, you'll learn enough to surprise the Agile know-it-alls at your office.

Better Backlogs

Chapters 13–15 dive deep into the lifecycle of a story. I'll discuss specific practices that help you use stories and story maps, starting with big opportunities and moving through the discovery work to identify a backlog full of stories that describe a viable product. You'll learn how

Documents
Document Sharing Hub
Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

story maps and lots of other practices can help you every step of the way.

Better Building

Chapters 16–18 dive deeper into using stories tactically, iteration-by-iteration or sprint-by-sprint. You'll learn how to get stories ready, to pay attention while they're built, to really get them done, and to really learn from each story you convert to working software.

I find that the last few chapters of many software development books are the extra junk. I can usually ignore them. Unfortunately, I didn't write any of those chapters. You'll need to read the whole book. My only consolation to you is that you'll get some useful nuggets out of every chapter that you can put to work right away.

Let's get to it.

Safari® Books Online



Safari Books Online is an on-demand digital library that delivers expert content in both book and video form from the world's leading authors in technology and business.

Technology professionals, software developers, web designers, and business and creative professionals use Safari Books Online as their primary resource for research, problem solving, learning, and certification training.

Safari Books Online offers a range of plans and pricing for enterprise, government, education, and individuals.

Members have access to thousands of books, training videos, and prepublication manuscripts in one fully searchable database from publishers like O'Reilly Media, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology, and hundreds more. For more information about Safari Books Online, please visit us online.

Documents
Document Sharing Hub
Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at *http://bit.ly/user-story-mapping*.

To comment or ask technical questions about this book, send email to *bookquestions@oreilly.com*.

For more information about our books, courses, conferences, and news, see our website at *http://www.oreilly.com*.

Find us on Facebook: *http://facebook.com/oreilly*

Follow us on Twitter: *http://twitter.com/oreillymedia*

Watch us on YouTube: *http://www.youtube.com/oreillymedia*

Documents
Document Sharing Hub
Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

Read This First

This book has no introduction.

Yes, you read that right. Now, you might immediately ask yourself, "Why doesn't Jeff's book have an introduction? Did he forget to write it? Is he beginning to slip after all these years?! Did the dog eat it?"

No, I didn't forget to write an introduction to this book. And, no, I'm not beginning to slip. At least I don't think I am. And my dog didn't eat it (although my daughter's guinea pig looks suspicious). It's just that I've long believed that authors spend too much time convincing me I should read their book, and a great deal of that convincing lives in the introduction. The meat of most books usually doesn't start until Chapter 3. And I'm sure it's not only me who does this, but I usually skip the introduction.

This book actually starts here.

And you're not allowed to skip this because it really *is* the most important part. In fact, if you only get two points from this book, I'll be happy. And those two points are right here in this chapter:

- The goal of using stories isn't to write better stories.
- The goal of product development isn't to make products.

Let me explain.

The Telephone Game

I'm sure you remember when you were a kid and you played this weird "telephone game" where you whispered something to somebody, who

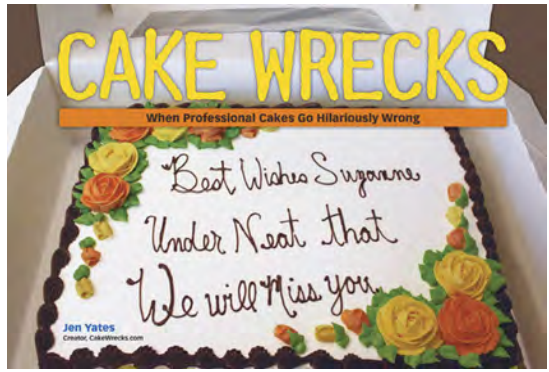
Documents
Document Sharing Hub

Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

whispered it to someone else, and so on around the group, until the last person reveals the totally garbled message and everyone laughs. Today, my family still plays this game at home with my kids around the dinner table. Note to parents: this is a good activity to occupy kids bored with adult dinner conversation.

In the grown-up world, we've continued this game—only we don't whisper to each other. We write lengthy documents and create official-looking presentations that we hand off to someone, who proceeds to get something completely different out of it than we intended. And that person uses that document to create more documents to give to different people. However, unlike that game we played as kids, we don't all laugh at the end.

When people read written instructions, they interpret them differently. If you find that a little hard to believe (it's in writing, after all!), then let me show you a few examples of instructions gone very, very wrong.



This is the cover of Jen Yates's book *Cake Wrecks* (Andrews McMeel Publishing). (Thanks to Jen and John Yates for supplying these.) The book sprang from her wildly entertaining website, *cakewrecks.com*. Please don't go there if you don't have at least an hour to waste. The site shows photos of oddly decorated cakes that defy explanation—but Jen explains them in spite of that. Now, one of the recurring themes in both the site and the book is misinterpreted requirements. But of course she doesn't refer to them as *requirements* because it's such a nerdy word. She calls them *literals* because the reader read and literally interpreted what was written. Looking at the photos, I can imagine someone listening to a customer and writing down what he wants, then handing that to someone else who'll decorate a cake.

Documents
Document Sharing Hub
Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

Customer: Hello, I'd like to order a cake.

Employee: Sure, what would you like written on it?

Customer: Could you write "So long, Alicia" in purple?

Employee: Sure.

Customer: And put stars around it?

Employee: No problem. I've written this up, and will hand it to my cake decorator right away. We'll have it for you in the morning.

This is the result:



Here's another. In software development, we call these *nonfunctional requirements*:



Documents
Document Sharing Hub
Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

These are funny examples, and we can laugh about wasting twenty bucks on a cake. But sometimes the stakes are much greater than that.

You've probably heard the story about the 1999 crash of a $125 million NASA Mars Climate Orbiter.[1] OK, maybe you haven't. But here's the punch line. If any project is sunk up to its eyeballs in requirements and written documentation, it's a NASA project. However, despite all the filing cabinets full of requirements and documentation, the orbiter crashed because while NASA used the metric system for its measurements, members of the Lockheed Martin engineering team used the old imperial measurement system to develop navigation commands for the vehicle's thrusters. While no one knows exactly where the orbiter ended up, some think it has found its happy place orbiting the sun somewhere past Mars.

Ironically, we put stuff in writing to communicate more clearly and to avoid risk of misunderstanding. But, way too often, the opposite is true.

Shared documents aren't shared understanding.

Stop for a minute and write that down. Write it on a sticky note and put it in your pocket. Consider getting it tattooed somewhere on your body so you can see it when you're getting ready for work in the morning. When you read it, it'll help you remember the stories I'm telling you now.

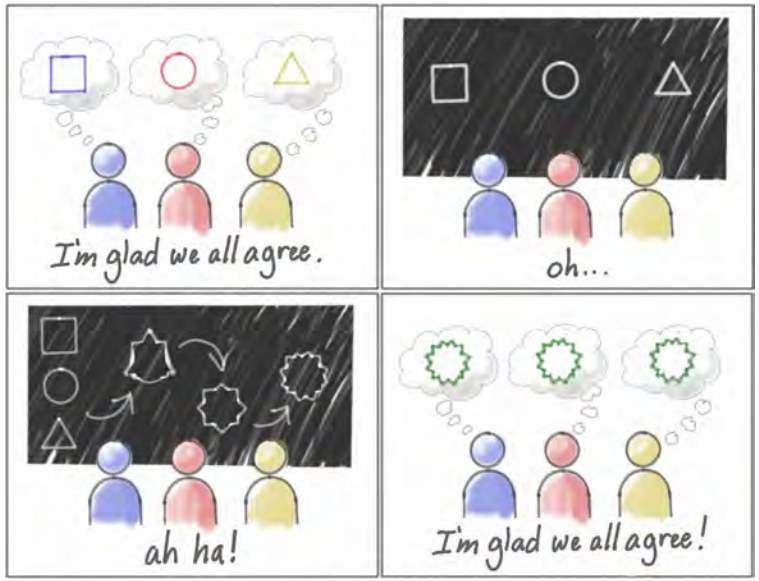
Shared understanding is when we both understand what the other person is imagining and why. Obviously, there wasn't shared understanding between several cake decorators and the people who gave them instructions in writing. And, at NASA, someone important didn't share understanding with others working on the guidance system. I'm sure if you've been involved in software development for a while, you don't have to reach back far in your memory to recall a situation where two people believed they were in agreement on a feature they wanted to add to the software, but later found out that the way one imagined it was wildly different from the other.

1. There are a lot of articles that try to describe what went wrong with the Mars Orbiter. Here's one of them: *http://www.cnn.com/TECH/space/9909/30/mars.metric.02/*.

Documents
Document Sharing Hub
Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

Building Shared Understanding Is Disruptively Simple

A former coworker of mine, Luke Barrett, first drew this cartoon to describe this problem. I asked him where he first saw it, but he didn't remember. So someone out there isn't getting the credit he or she deserves. For years I saw Luke step through these four frames as slides in a PowerPoint deck while I casually dismissed them as interesting but obvious. Apparently I've got a thick head. It's taken me many years to understand how this cartoon illustrates the most important thing about using stories in software development.



The idea is that if I have an idea in my head and I describe it in writing, when *you* read that document, you might quite possibly imagine something different. We could even ask everyone, "Do you all agree with what's written there?" and we might all say, "Yes! Yes, we do."

However, if we get together and talk, you can tell me what you think and I can ask questions. The talking goes better if we can externalize our thinking by drawing pictures or organizing our ideas using index cards or sticky notes. If we give each other time to explain our thoughts with words and pictures, we build shared understanding. It's at this

Documents
Document Sharing Hub
Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

point, though, that we realize that we all understood things differently. That sucks. But at least now we know.

It's not that one person is right or wrong, but that we all see different and important aspects. Through combining and refining our different ideas, we end up with a common understanding that includes all our best ideas. That's why externalizing our ideas is so important. We can redraw sketches or move sticky notes around, and the cool thing is that we're really moving ideas around. What we're really doing is evolving our shared understanding. That's super-hard with just words alone.

When we leave this conversation, we may still name the same feature or enhancement, it's just that now we actually mean the same thing. We feel aligned and confident we're moving forward together. That's the quality we're managing to. And, sadly, it's intangible. You can't see or touch "shared understanding," but you can feel it.

Stop Trying to Write Perfect Documents

There are a great number of people who believe that there's some ideal way to document—that, when people read documents and come away with different understandings, it's either the reader's fault or the document writer's. In reality, it's neither.

The answer is just to stop it.

Stop trying to write the perfect document.

Go ahead and write something, anything. Then use productive conversations with words and pictures to build shared understanding.

The real goal of using stories is shared understanding.

Stories in Agile development get their name from how they should be used, not what you write down. If you're using stories in development and you're not talking together using words and pictures, you're doing it wrong.

If your goal in reading this book is to learn to write better stories, you've got the wrong goal.

Documents
Document Sharing Hub

Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

Good Documents Are Like Vacation Photos

If I show you one of my vacation photos, you might see my kids on a beach and politely say, "That's cute," but when I look at my vacation photo, I remember a particular beach in Hawaii that we had to drive more than an hour on a deeply rutted four-wheel-drive trail, and then hike another half-hour over lava fields to get to. I remember my kids whining, saying nothing could possibly be worth this, and me wondering the same thing. But it was. We enjoyed a blissful day on an incredible beach where very few people were, which is why we took the trouble to get there. The turtles came up on the shore to bask on the sand, which was the icing on the cake of this fabulous day.



Of course, if you look at the picture you won't know all that because you weren't there. I remember all that because I was.

For better or worse, this is the way documents actually work.

If you participate in lots of discussions about what software to build, and then create a document to make sense of it, you might share it with someone else who was there. You might both agree it's good. But remember, your shared understanding is filling in details that aren't in the document. Another reader who wasn't there won't get the same things from it that you will. Even if she says she gets it, don't believe

Documents
Document Sharing Hub
Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

her. Get together and use the document to tell a story the same way I used my vacation photo to tell you my story.

Document to Help Remember

I've heard people joke, "We're using an Agile process because we've stopped writing documents." It's a joke for people who know, because a story-driven process needs lots of documents to work. But those documents don't always look at all like traditional requirements documents.

It takes talking and sketching and writing and working with sticky notes or index cards. It's pointing to documents we brought into the conversation and marking them up with highlighter and scribbled notes. It's interactive and high energy. If you're sitting at a conference table while a single person types what you say into a story management system, you're probably doing it wrong.

When you're telling stories, most anything can be used as a tool to communicate. And as we tell these stories, and write lots of notes, and draw lots of pictures, we need to keep them. We carry them around to look at later, photograph, and retype into more documents.



Documents
Document Sharing Hub
Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

But, remember, what's most important isn't what's written down—it's what we remember when we read it. That's the vacation photo factor.

Talk, sketch, write, use sticky notes and cards, and then photograph your results. Even better, shoot a short video of you talking through what's on the board. You'll remember lots of details in a remarkable depth that you could never possibly document.

To help remember, photograph and shoot short videos of the results of your conversations.

Talking About the Right Thing

There are lots of people who believe their job is collecting and communicating requirements. But it's not.

The truth is, your job is to change the world.

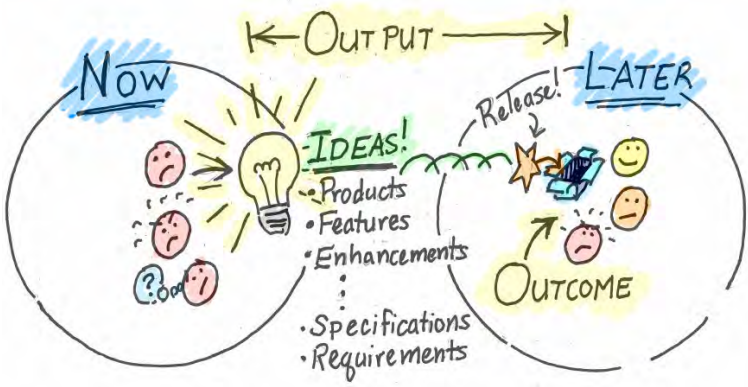
Yes, I said that to get your attention. And, yes, I know that sounds like hyperbole. That's because that phrase is usually associated with world peace, eliminating poverty, or even more far-fetched goals like getting politicians to agree with one another. But I'm serious. Every great idea you turn into a product solution changes the world in some small, or not-so-small, way for the people who use it. In fact, if it doesn't, you've failed.

Now and Later

There's a simple, change-the-world model that I personally use and keep in my head all the time, and you need to keep it in your head too while you're having story conversation and building shared understanding.

Documents
Document Sharing Hub
Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

I draw the model like this:



The model starts by looking at the world as it is now. When you look at the world as it is now, you're going to find people who are unhappy, mad, confused, or frustrated. Now, the world's a big place, so we'll focus mostly on the people who use the software we make, or the people we hope will use it. When you take a look at what they're doing—and the tools they use and how they're doing things—you're going to come up with ideas, and the ideas might be for:

- Entirely new products you can build
- Features to add to an existing product
- Enhancements to products that you've built

At some point in time, you'll have to communicate details about your ideas to some other people, and you might start to do some design and specification. If you're going to hand all this off to someone else, then you might indeed call all these details your *requirements*. But it's important to remember that requirements are just another name for the ideas we have that would help people.

Given those requirements, we go through some process that results in a delivery, and out comes some software that actually lands in the world, and it lands in the world *later*. And what we hope is true is that those people who were initially unhappy, mad, frustrated, or confused will become happy when that software lands. Now, they're not happy because they saw the pretty box it came in—software doesn't usually come in boxes these days anyway. They're not happy because they read

Documents
Document Sharing Hub
Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

the release notes, or downloaded the app to their mobile device. They're happy because when they use the software, or the website, or the mobile app, or whatever you've built, they do things differently—and that's what makes them happy.

Now, the truth is that you can't please everyone all of the time. Your mother should have told you that. Some people will be happier than others with whatever it is that you produce, and some might still be unhappy no matter how hard you've worked and how amazing your product might be.

Software Isn't the Point

Everything between the idea and the delivery is called *output*. It's what we build. People working in Agile software development will deliberately measure *velocity* of output and try to speed up their rate of output. As people are building software, they are, of course, concerned about the cost of what they're doing and the speed at which what they're doing gets done, as they should be.

But, while it's necessary, the output isn't the real point; it's not the output that we really wanted. It's what comes after as a result of that. It's called *outcome*. Outcome is what happens when things come out—that's why it's called that—and it's difficult because we don't get to measure outcome until things do come out. And we don't measure outcome by the number of features delivered, or what people have the capability to now do. We measure what people actually do differently to reach their goals as a consequence of what you've built, and most important, whether you've made their lives better.[2]

That's it. You've changed the world.

You've put something in it that changes the way people can reach their goals, and when they use it, the world is different for them.

If you remember, your goal isn't to just build a new product or feature. When you have conversations about that feature, you'll talk about who it's for, what they do now, and how things will change for them later. That positive change later is really why they'd want it.

2. The clean language and distinction between *output* and *outcome* was first made clear to me in a talk by Robert Fabricant called "Behavior Is Our Medium". Prior to that, I'd struggled with language that was clear in my head—and everyone else's too. Happily, it was clear in Robert's head.

Documents
Document Sharing Hub
Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

Good story conversations are about who and why, not just what.

OK, It's Not Just About People

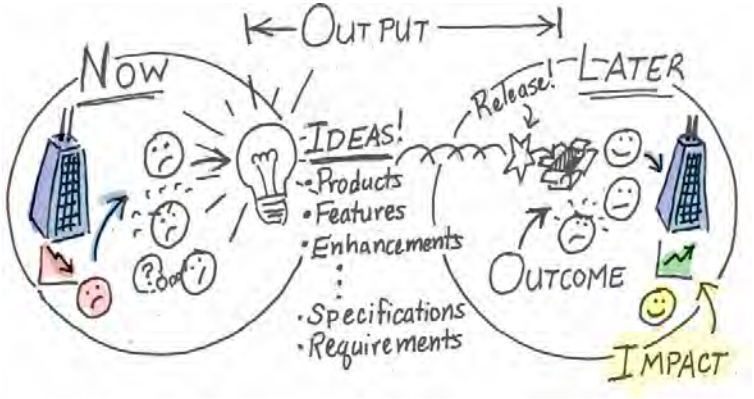
I care about people as much as the next guy, but truthfully, it's not just about making people happy. If you work for a company that pays you and others, you've got to focus on what ultimately helps your organization earn more, protect or expand its market, or operate more efficiently. Because, if your company isn't healthy, then you won't have the resources (or the job) to help anyone.

So I've got to revise this model a bit. It actually starts by looking inside your organization. There you'll find even more people who aren't happy. And it's usually because the business isn't performing as well as they'd hope. To fix this, they may have ideas to focus on specific customers or users and to make or improve the software products they're using. You see, it ultimately is about people, because:

Your company can't get what it wants unless your customers and users get something they want.

The flow continues by choosing the people to focus on, the problems to solve, and the ideas to turn into working software. And from there—if the customers buy, and the users use it, and people are happy—eventually the business that sponsored this development will see the benefit it's looking for. That'll be reflected in things like increased revenue, lower operational costs, happier customers, or expanded market share. This makes lots of people inside your company happy. It should make you happy, too, since you've just helped your company stay healthy while making real people's lives better in the process. It's a win-win.

Documents
Document Sharing Hub
Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...



It's that longer-term stuff that happens as a consequence of good outcomes that's I'll label *impact*. Outcomes are often something you can observe right away after delivery. But impact takes longer.

Build Less

There's an uncomfortable truth about the software world, and I suspect it's true of lots of other places. But I know software. And what I know is that:

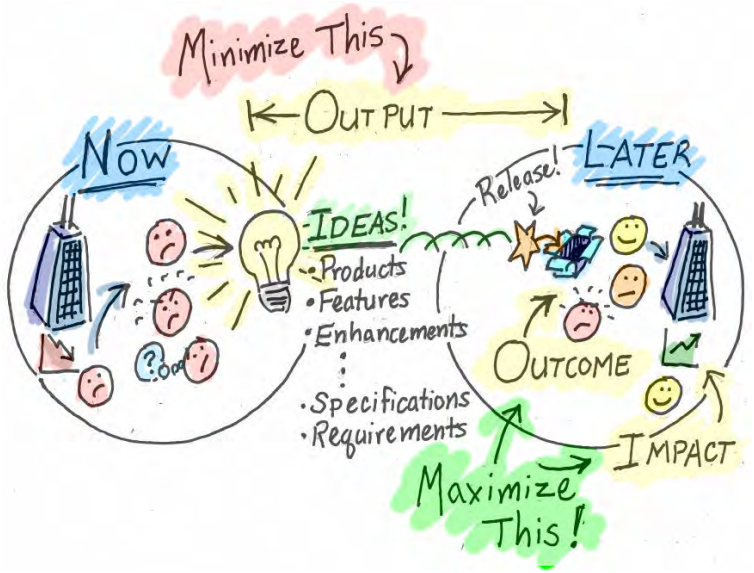
There's always more to build than we have time or resources to build—always.

One of the common misconceptions in software development is that we're trying to get more output faster. Because it would make sense that if there was too much to do, doing it faster would help, right? But if you get the game right, you will realize that your job is not to build *more*—it's to build *less*.

Minimize output, and maximize outcome and impact.

At the end of the day, your job is to *minimize* output, and *maximize* outcome and impact. The trick is that you've got to pay close attention to the people whose problems you're trying to solve. These include the people who will choose to buy the software to solve a problem in their organizations, the *choosers*, as well as the people who use it, the *users*. Sometimes they're the same people. Sometimes they're not.

Documents
Document Sharing Hub
Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...



Your business has lots of possible users and customers it could focus on. Your businesses strategy should give you some guidance about who to focus on to get the impact you want. I promise you that no business has the resources to make *everyone* happy—it's just not possible.

Don't get me wrong here. Building more software faster is always a good idea. But it's never the solution.

More on the Dreaded "R" Word

For almost the entire first decade of my software career, which I spent building software for brick-and-mortar retailers, I got away without using the word *requirements*—at least, not much. It just wasn't a relevant term for what I was doing. I had lots of different customers who all had specific ideas about what would help them. I also knew I worked for a company that had to make money by selling my product. In fact, I'd spent long hours standing at trade shows helping my company sell its product to a wide variety of customers. I knew at the end of the day that I would have to continue to work with those customers after I shipped the products my team and I developed, and so I diligently worked to act in their best interest. This meant I couldn't give everyone everything they wanted, because they wanted different things. And my

Documents
Document Sharing Hub
Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

company and team didn't have infinite time, so I had to work hard to figure out the least I could build to make people happy. That may sound frustrating, but it's actually the fun part.

As the company grew, we added more traditional software people. At one point, the head of a different team came to me and said, "Jeff, I need you to make these changes to the product you're working on."

I said, "Great, no problem. Tell me who they're for and what problems this solves for them."

Her response? "They're the requirements."

I replied, "I get it. Just tell me a bit about who they're for, and how they're going to use this, and where it fits into the way they work."

She looked at me like I was stupid and said to me one last time with an air of finality, "They're *requirements*."

It was at that moment that I learned that the word *requirements* actually means *shut up*.

For a great many people, that's exactly what requirements do. They stop conversations about people and the problems we're solving. The truth is, if you build a fraction of what's required you can still make people very happy.[3]

Remember: at the end of the day, your job isn't to get the requirements right—your job is to change the world.

That's All There Is to It

If you get nothing else from this book, remember these things:

- Stories aren't a written form of requirements; telling stories through collaboration with words and pictures is a mechanism that builds shared understanding.
- Stories aren't the requirements; they're discussions about solving problems for our organization, our customers, and our users that lead to agreements on what to build.

3. Because I strongly agree with the sentiment, I'm paraphrasing the way Kent Beck cautions against the misuse of the term requirement in his book *Extreme Programming Explained* (Addison-Wesley).

Documents
Document Sharing Hub
Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

- Your job isn't to build more software faster: it's to maximize the outcome and impact you get from what you choose to build.

Stories as they're intended are a completely different way of thinking about the challenges we face working together to create software—and lots of other things, for that matter. If you can work together effectively and create things that solve problems, you will rule the world. Or at least some small part of it inhabited by your products.

As you read this book, my hope is that you get back to the basics of using stories. I hope you work together with others, telling stories about your users and customers and how you can help them. I hope you draw pictures, and build big sticky-note models. I hope you feel engaged and creative. I hope you feel like you're making a difference. Because when you do it right, you are. And it's a lot more fun, too.

Now it's time to talk about the most fun you can possibly have telling stories, and that's when you're using a story map.

Documents
Document Sharing Hub

Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

CHAPTER 1

The Big Picture

"I love Agile development! Every few weeks we see more working software. But it feels like I've lost the big picture."

If I had a dime for every time I heard something like that from an Agile team member, I'd have…well…a lot of dimes. I hear it a lot. You may have even said something like that yourself. Well, I've got good news for you. Using an Agile process and a story-driven approach doesn't mean you have to sacrifice the big picture. You can still have healthy discussions about your whole product and still see progress every few weeks.

Since you've patiently read the "Read This First" chapter, I'm going to bypass all the junk about stories and proceed directly to how story maps solve one of the biggest problems in Agile development. If you're already familiar with writing stories on Agile projects, this chapter may be enough to get you started.

The "A" Word

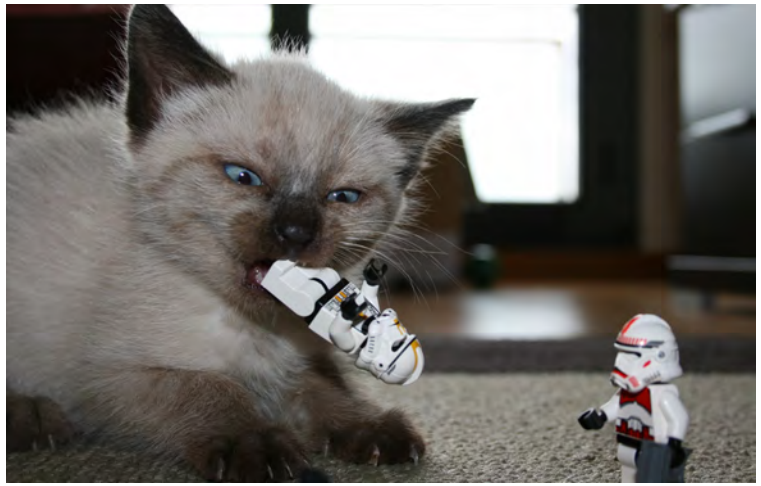
If you're reading this book, you likely know that story mapping is a way to work with user stories as they're used in Agile processes. Now, it's at this point that every other book that has something to do with Agile development reproduces the "Manifesto for Agile Software Development," that thing written in 2001 by 17 guys who were frustrated with some of the big counterproductive process trends going on at the time. I'm glad they wrote it. And I'm glad that the impact of their work has been felt by so many.

Documents
Document Sharing Hub

Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

But I'm sorry to disappoint you—I'm not going to reprint the manifesto and gush about why it matters. I believe you already know why it does. And, if you haven't read the manifesto, then you should.

In the space that the manifesto would have taken up in this chapter, I am instead including a funny kitten photo.[1] Why? Because it has been proven time and time again that funny kitten photos on the Internet get far more attention than *any* manifesto could ever hope to.



So, you might wonder, what does this kitten have to do with Agile? Actually, nothing. But Agile definitely has something to do with this book, and with stories and the evolution of story mapping.

<Cue the flashback music…>

I was working at a startup in San Francisco in 2000, and the company had hired Kent Beck (the guy who created Extreme Programming and first described the idea of stories) as a consultant to get the software development process going. I'm rewinding way back, but the important thing is this story idea is an old one. If you're just starting out with using stories, you lost any early adopter status you could have had a decade or so ago. Kent and others who pioneered Extreme Programming knew that all those ways of doing requirements in the past didn't work out well. Kent's simple idea was that we should get together and

1. Photo taken by Piutus, found on Flickr and licensed under the Creative Common Attribution license.

Documents
Document Sharing Hub
Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

tell our stories; that by talking we could build shared understanding, and together we'd arrive at better solutions.

Telling Stories, Not Writing Stories

When I first heard the term *story*, it bugged me. I'll admit it. The idea that we'd trivialize the important things that people wanted by calling them stories didn't seem right. But I'm a slow learner—a point I brought up earlier when discussing shared understanding. It took me a while to really get that:

> *Stories get their name from how they should be used, not what should be written.*

Even before I'd really understood why stories had that name, I realized that I could write down a bunch of stories—a sentence or a short title—on sticky notes or cards. I could move them around and prioritize them to decide which one was more important. Once I decided that one was more important than another, then we could start having a discussion about it. This was super-cool. Why hadn't I ever written things on cards and organized them this way before?

The problem was that this one card could be something that might take a software developer just a couple hours to add to a product, or maybe a couple days or a couple weeks, or maybe a month—who knew? I didn't—at least not until we started talking about it.

I got into a nasty argument while working with stories on my very first Agile project when I began a story conversation and learned that my story was too big. I'd hoped to get this story done in the next iteration. The developers I spoke with informed me otherwise. I felt like I'd done something wrong. The developers identified a small part we could talk about that could be accomplished in our next iteration. But I left frustrated that we couldn't talk about the big picture. I really wanted to understand how much time the big thing I really needed would take. I'd hoped this discussion would accomplish that, and it didn't.

Telling the Whole Story

In 2001 I left the team I was on and started doing things differently. I, and my team, tried an approach to writing stories that focused on the big picture. We worked to understand the product we were building

Documents
Document Sharing Hub
Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

and to make tradeoffs together. We used that bunch of index cards with story titles to organize our thoughts and break down that big picture into the small parts we could build next. In 2004, I wrote my first article about this idea. I didn't coin the term *story mapping*, however, until 2007.

It turns out that the name you give something matters. It was after giving the practice a good name that I really saw it spread. I thought it was a great invention at the time—that is, until I started running into more people who were doing similar if not exactly the same things. I'd discovered a *pattern*.

I first heard this definition of a pattern from my friend Linda Rising: when you tell someone about a great idea and he says, "Yeah, we do something like that, too." It's not an invention, it's a pattern.

Story mapping is a pattern. It's what sensible people do to make sense of a whole product or whole feature. It's what they do to break down large stories into smaller ones. Don't feel bad if you didn't arrive at it on your own. You would have eventually. But reading this book will save you weeks or months of frustration.

Story maps are for breaking down big stories as you tell them.

Today, company after company has adopted the idea of story mapping. My friend Martina at SAP said in a message she sent in September 2013 that:

> …at this point more than 120 USM [User Story Mapping] workshops have officially been recorded. A lot of POs just simply love it! It is simply a well-established approach at SAP.

Every week I hear from someone else from somewhere else telling me how mapping stories helped solve a problem for them. These days, I learn more from talking to others than I ever could on my own.

The original idea of stories was a simple one. It turned our focus away from shared documents and toward shared understanding. A common way to use stories is to build a list of them, prioritize them, and begin talking about them and then turning them into software one at a time. That sounds pretty reasonable when you hear about it. But it can create some big problems.

Documents
Document Sharing Hub

Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

Gary and the Tragedy of the Flat Backlog

A few years ago I met Gary Levitt. Gary was a businessperson in the process of launching a new web product. The web product is out there right now, and it's called Mad Mimi, which when Gary conceived of his product, was short for *music industry marketing interface*.[2] Gary is a musician who had his own band. He managed his band, helped manage others, and was also a studio musician and created recordings for clients.



The day I met Gary he had an order from the Oprah Winfrey show for dozens of intros and outros, little bits of music that are used to go out to and come in from commercials and things like that. Producers of television shows buy those the way people laying out a newsletter buy clip art, so it's like audio clip art. Gary had an idea for a fairly big application that would help musicians like him and people he knew to collaborate with one another on projects like the one he was working on, along with lots of other things a band manager and musician would need to do to manage and promote his band.

Gary wanted to get the software built so he worked with somebody, and that somebody was working in an Agile way. That person told Gary to write down a list of all the things he wanted, prioritize the list, and then they would talk about the highest-valued things—the most important—and start building them one at a time. That list of things

2. Read about Gary in the *Business Insider* article "How This Guy Launched A Multi-Million Dollar Startup Without Any VC Money".

Documents
Document Sharing Hub
Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

is what Agile processes refer to as a *backlog*, and it seemed to make sense to Gary to create the list and start with the most important things first. So that's what he did.

Gary created his backlog and the development team started building things a bit at a time. In the meantime, Gary was hemorrhaging cash as he continued to pay for each piece of software that was built. The software was slowly taking shape, but Gary could tell it was going to take a lot longer for it to match his vision and he was going to run out of cash long before then.

I knew the person who was working with Gary. My friend knew Gary was stressing out and wanted to help him. The somebody I knew asked if I could have a conversation with Gary, to talk with him and help him get his ideas organized. I contacted Gary and made arrangements to meet him at his office in Manhattan.

Talk and Doc

Gary and I started talking. And as he talked, I wrote cards with key points from what he said. There's a mantra that I like when I build story maps. I'll say "talk and doc" (short for the verb *document*), which basically means don't let your words vaporize. Write them down on cards so you can refer back to them later. You'll notice how pointing to a few words on a card quickly helps everyone recall the conversation about it. We can slide them around the table where we can reorganize them. We start using useful words like *this* and *that* as we point to cards. It saves lots of time. Helping Gary externalize his thoughts was critical to getting shared understanding. It wasn't a habit for him, so it was easy for me to write the cards as he told the story.

> *Talk and doc: write cards or sticky notes to externalize your thinking as you tell stories.*

We started by placing cards on a tabletop, but quickly ran out of space. Gary was moving offices the day I visited with him, and much of the furniture in the New York City loft where he was located was off the floor. So we moved our growing map of cards onto the floor.

Documents
Document Sharing Hub

Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

At the end of the day, the floor looked like this:



Think — Write — Explain — Place

When working with a team to build a story map, or having discussions about anything, create a simple visualization to support your discussion. One of the things that goes wrong is lots of ideas *vaporize*—that is, we say them, and people nod as if they've heard. The ideas are not written down or referred to. Then, later in the conversation, the ideas come up again and unfortunately need to be re-explained because people didn't really hear or forgot them.

Get in the habit of writing down a little about your idea before explaining it.

1. If you're using cards or sticky notes, *write* down a few words about your idea immediately *after thinking* it.
2. *Explain* your idea to others as you point to the sticky note or card. Use big gestures. Draw more pictures. Tell stories.
3. *Place* the card or sticky into a shared workspace where everyone can see, point to, add to, and move it around. Hopefully, there will be lots of other ideas from you and others in this growing pile.

I find that when I'm doing my best to listen to others, what they're saying causes me to think of other ideas. I used to try to hold those ideas in my head and wait for a moment to inject them into the conversation, resorting to outright interruption if the time didn't come soon enough. But then I realized I'd stopped listening to the person

Talk and Doc Documents
Document Sharing Hub
Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...

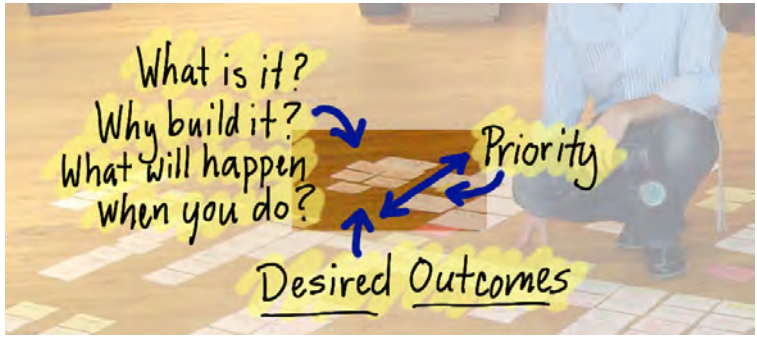
who was talking, as my limited brainpower was focused on recalling my great idea. Today, I simply scribble the idea on a sticky note and set it aside to wait for a better point in the conversation to inject it. Somehow writing it pops it out of my head so I can focus on what I'm hearing. And reading it from the sticky later helps me recall my idea and explain it.

I wasn't here to capture Gary's requirements. And the first thing we talked about wasn't that list of features. We had to back up a bit and start at the beginning.

Frame Your Idea

Our first conversation focused on framing his product idea. We talked about his business and what his goals were. *Why are you building this? Tell me about the benefits for you and for the people who will use this. What problems does it solve for those people and for you?* As you read this you might detect I've got that now-and-later model in my head. I'm trying to understand the outcomes Gary is looking for, not the output he wants to build.

If I put two cards down, one above the other, then people assume that the one above is more important. Without saying a word, if I simply slide a card above another, I've indicated something about importance. Try that with a list of goals. Purposely put them in the wrong order and watch the person you're working with reach out to adjust them. I did this with Gary and his goals, and it helped him express what was more important to him.



Documents
Document Sharing Hub
Full document details: User Story Mapping Discover the Whole Story, Build the Right Product (Jeff Patton,...