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

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

Want to read more?

You can buy this book at **oreilly.com**
in print and ebook format.

Buy 2 books, get the 3rd FREE!
Use discount code: OPC10
All orders over $29.95 qualify for **free shipping** within the US.

It's also available at your favorite book retailer,
including the iBookstore, the Android Marketplace,
and Amazon.com.

O'REILLY®

Spreading the knowledge of innovators

oreilly.com

User Story Mapping

Jeff Patton

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

O'REILLY®

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]

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

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

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

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

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

References 265

Index 267

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.

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.

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

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.

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".

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.

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

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.

Describe Your Customers and Users

Gary and I continued to talk and doc. The next conversation Gary and I had was about the customers who would buy, and users who would use, his piece of software. We listed the different types of users. We talked about what benefits they would get, and asked why they would use the product and what we thought they would do with it. What was in it for them? We built a big pile of those. The cards naturally seemed to fall with most important users higher in the pile. Funny how it works out that way without an explicit decision.

Before we'd gone into any detail at all, I could already see that Gary's vision was big. One of the tough realities about software development is that there's always more to build than we have time and money for. So the goal should *never* be to build it all. The goal is to minimize the amount we build. So the first question I asked Gary was, "Of all these different users and the things they want to do, if we were to focus on thrilling just one of those users, who would it be?"

Gary chose one and we started to really tell stories.

Mad Mimi User Types

These are the different types of users Gary described for Mad Mimi. Just naming them and writing a little about what they want helped us both see that there was a lot here. Even before discussing features, we'd decided to defer creating software for some types of users.

Tell Your Users' Stories

I next said, "OK, let's imagine the future. Let's assume for a minute this product is live and let's talk about a day in the life of someone who uses it and start telling the story. First, they would do this, and then this, and so on and so on." And we told the story in a flow from left to right. Sometimes we backtracked and put things to the left of other things, and because they were on cards, we could easily rearrange them.

The other interesting thing that happens naturally when working with cards is if I put one to the left and another to the right, without saying a word I've indicated sequence. This is kind of magical for me—but I'm easily entertained. I marvel at how much we can communicate without saying a word.

Reorganizing cards together allows you to communicate without saying a word.

As we talk and doc, and as I write down our conversation, we're building something really important. No, it's not that pile of cards on the floor. The something that's really important is *shared understanding*. We're getting on the same page. This is something Gary had never done with anyone before about his product idea, at least not at this level of detail. He'd never even given it this much thought himself. The high points were in his head, sort of like the action scenes you'd see in a movie preview.

Before now, Gary had done what he was asked to do. He'd written a bunch of story titles, put them in a list, and talked about them one at a time. The conversations were more about the details of what to build and less about this big picture. And there were a lot of holes in Gary's big picture. You'll find that no matter how clear you are about your story, talking through it while you map will help you discover the holes in your own thinking.

Mapping your story helps you find holes in your thinking.

As we dug deeper, we realized that the story also wasn't just about one user. Gary's started with a band manager who wanted to promote his band and the work he was doing to create the promotion and email it to fans. Then we quickly had to talk about the fan of the band, and tell her story about seeing the promotion and then making plans to see a show.

Then, if we were promoting the band someplace, we'd need to tell the story of the venue's manager and the information he'd like to learn about the promotion. By this time, our map was wide enough that we bumped into the wall, so we had to continue the story in another layer below the first. That's why the map in the photograph has two layers.

During the story, sometimes Gary would get to a part where he was excited and he'd start describing lots of details. One card above another can indicate priority. But it can also mean *decomposition*, which is just a fancy word for smaller details that are part of a bigger thing. As Gary described the details, I'd record them on a card, and place them below the big user step above. For instance, when Gary described creating the flyer that band managers would use to promote their gigs, he was extra passionate and had lots of details to discuss.

Gary lived in New York City, and when bands are composing flyers he's imagining all these really cool things he sees stuck on walls and lampposts in New York. They might look like they were put together with glue and tape and then photocopied, but some were really elegant and artistic. After recording a handful of details, I said, "Let's come back and get to the details later. Let's continue on and move this story forward." It's easy to get lost in the details, especially the ones you're passionate about. But, when we're trying to get the big picture, it's important to get to the end of the story before catching all those details. Another mantra I use when mapping, at least at this stage, is "think mile wide, inch deep"—or for people in sane countries using the metric system: "kilometer wide, centimeter deep." Get to the end of the story before getting lost in the details.

Focus on the breadth of the story before diving into the depth.

Eventually we *did* get to the end of Gary's story. The band manager had successfully promoted a gig to thousands of fans who spread the word, and the show was a wild success. The product vision so far was clear in both our heads. I said, "Now let's go back and fill in the details and consider some of the alternatives."

Mimi's Big Story

If you read across the top of Gary's map, you'll see big activities like:

- Signing up
- Changing my service
- Viewing my band stats
- Working with my show calendar
- Working with my audience
- Publicizing a show
- Signing up for a band's email list
- Viewing promotions online

There were lots of other big things at the top of the map, but that's a good subset to give you an idea of what you'd write on a card. Notice how we can assume who does what. When Gary said, "Publicizing a show," he knew he was talking about the band manager. When I said, "signing up for the band's email list," Gary knew I was talking about the band's fan. Those cards were close by and easy to point to during our conversation.

"Publicizing a show" was a big thing. It broke down into these steps arranged left to right underneath the "Publicizing the show" card.

- Start a show promotion.
- Review the promo flyer Mimi created for me.
- Customize the promo flyer.
- Preview the promo flyer I created.

Notice how what we wrote on every card are short verb phrases that say what the specific type of user wants to do. Writing them this way helped us tell the story: "the band manager would then publicize the show. To do that he'd start a promotion, then review the flyer Mimi created, then customize it, and then…" Notice how when you put "and then" in between what's written on each card, you get a nice story.

Explore Details and Options

After we've got the breadth of the story map in place, it starts thickening up. The cards we put at the top of each of the columns in the map become big things, and then the details break down below them. We stop at each step in the user's story and ask:

- What are the specific things they'd do here?
- What are alternative things they could do?
- What would make it really cool?
- What about when things go wrong?

At the end of this we'd gone back and filled in a lot of details. The result was that we had told the story about a day in the life of a band manager, as well as the other people important to the band manager's success: fans and venue managers.

The Details

If you look inside a story step like "Customize the promo flyer," you'd see details like:

- Upload an image
- Attach an audio file
- Embed a video
- Add free text
- Change the layout
- Start with a promotion I've used before

You can see that even these smaller steps will need a lot more discussion to work out the details. But at least we could begin to name them all.

Notice how what's written on the cards are also those short verb phrases that help you tell stories. We can string this together with phrases like "or he might" like this: "to customize the flyer the band manager might upload an image, or he might attach an audio file or embed a video, or…" It's pretty cool, really.

I asked Gary, "Now what? We have all these other users with other things they want to do—do you want to talk about them? You can see that if we keep talking we'll need a bigger room. And, Gary, if you do all this stuff, it will take a lot of money to build this product. We could talk about the rest of this stuff, but if we built this much and you launched your product and just did this, that looks like it'd be a valuable product."

Gary agreed, and he said, "I'm going to stop there."

The sad part of this story is that I asked Gary, "You've been building a lot of software so far, but how much of the software you've built is on this map we've created?"

"Nearly none of it," Gary replied, "because when I built a list and prioritized things, I sort of assumed we needed to start somewhere else. I was thinking about the whole big vision of this thing, the vision that would have taken me years to reach, and now that we've had this discussion I wouldn't have started there at all."

Story mapping is all about having a good old-fashioned conversation and then organizing it in the form of a map. The part that most people look at is the map—that left-to-right shape with the steps people take to tell a big story. The top to bottom is about the details. But the critical parts that frame the product and give more context are often hung above and around the map. They're the product's goals, and information about its customers and users. If you keep a map on the wall, you'll find it's good idea to add user interface (UI) sketches and other notes around the map.

In just a day working together, Gary and I built shared understanding around the product he wanted to build. But there was a storm cloud forming above our heads, and we knew it. Inside each of those cards we wrote were lots of details and lots more discussions. And, for Gary, all those details and all those discussions equated to money he would need to spend to build software—money he didn't have. He'd learned one of the fundamental truths about software development: there's always more to build than you'll have time for.

Now, there are a lot of other big assumptions Gary was making about the people who'd use his product, and if they really wanted to, or really could use it as he envisioned. But, right now, those things weren't his biggest concern. He needed to work harder to minimize his product idea to something that was feasible first.

Gary's story eventually has a very happy ending. But in the next chapter I'll tell the story of another organization that learned it had way too much to build, and how it used a map to find a viable solution.

Artgility—Creativity in Art Meets Creativity in IT

Ceedee (Clare) Doyle, Agile Project Manager and Coach, Assurity Ltd, Wellington, NZ

Background

The Learning Connexion (TLC) is an art college in Wellington, New Zealand, that teaches art and creativity. TLC's programs are unique because they are based on "learning by doing"; that is, the practice is the theory. In conjunction with tutors, students develop briefs that connect with the ideas they choose to explore.

TLC was a typical small-to-medium-size organization that had developed ad hoc IT systems to support the needs it had at the time. Student information was collected in five different places and was different in each! TLC needed some way to manage students that would work for it and the way it teaches, which is quite different from most educational establishments.

TLC had no experience with IT projects. Each small application that had been built for it had been done by somebody's-brother's-friend's-flatmate's-dog using simple technologies like Microsoft Excel and Access. The sole commercial application (used for statutory reporting) double-handled data from the other four sources.

As a former student, I had kept in touch with the team and when they needed some help they contacted me. In 2009 I had been in IT for nine years and had wanted to do an Agile project for the last three, ever since I had heard about it. This was the right place, the right project, and the right time to do it!

Project Phoenix

The initial workshops were going to be two half-day sessions with key staff members. I was working with a large, diverse group, and my goal was to develop shared understanding. I started with an overview of how story mapping works and an overview of the big steps in the school's student management process.

Up until I showed them this picture (the backbone of the story map), the team members each had an idea of what *they* did, but, as Alice the sponsor said, it was probably the first time they all had a clear picture of their own business process and how all the steps interacted.

From there we brainstormed what people wanted the system to do. The scope was *massive*, and the stories were *many*.

The beauty was that these were creative people and they were used to the "appreciative enquiry" method, so braindumping everything they could think of that the system needed to do was something they took to like a baker makes bread.

The main headings (from the diagram) were Enquiries → Admissions → Enrollments → Classes → Complete work → Completion → Graduation.

Using the story mapping guidelines, we then walked through each section to make sure that it made sense, and got the flow for a student through each step of the process. Several people had lights go on suddenly, as they realized where they fit into the overall process and *why* they had to do some of their activities, and others realized they were being left out of certain steps that would make a big difference to them. Stepping through the story map and my emphasis on having stories vertically—which happened together—showed places where they could work together better and steps that were doubled up. Until this point, the team had little view of what everyone else was doing, but they suddenly developed shared understanding of how the whole process worked and a common lexicon. In one example, *Classes* was renamed *Delivery* because not all students attend classes.

When it came to prioritizing, it wouldn't have worked to identify the must-haves, should-haves, and could-haves—it was either in or out. It was very simple: "we cannot go live without this" above the line, and everything else below it. After we'd walked through the Enquiries, the team got it and spent the remainder of the day doing the rest. I was able to step out of it! The staff took over and started adding subheadings to better describe that "these things all have to happen together, and then these things." So, by the end, they had created, as a group, a big picture of the steps a student follows to get from initial enquiry to graduation.

What started out as two half-day sessions turned into three full days of workshops in which people came and went as the need arose (they had to teach classes, and so forth). The flexibility of the workplace meant that nearly all the staff came through the Phoenix room and had their two cents' worth. They found the process really useful to get the big picture and for everyone to have their wants included. It also identified where there were gaps and made it easy to sift out what was really vital. At the end of it, we had a clear picture of what was to be in the first cut of the software.

O'Reilly Ebooks—Your bookshelf on your devices!

When you buy an ebook through oreilly.com you get lifetime access to the book, and whenever possible we provide it to you in five, DRM-free file formats—PDF, .epub, Kindle-compatible .mobi, Android .apk, and DAISY—that you can use on the devices of your choice. Our ebook files are fully searchable, and you can cut-and-paste and print them. We also alert you when we've updated the files with corrections and additions.

Learn more at ebooks.oreilly.com

You can also purchase O'Reilly ebooks through the iBookstore, the Android Marketplace, and Amazon.com.

O'REILLY®

Spreading the knowledge of innovators

oreilly.com