Extracted from:

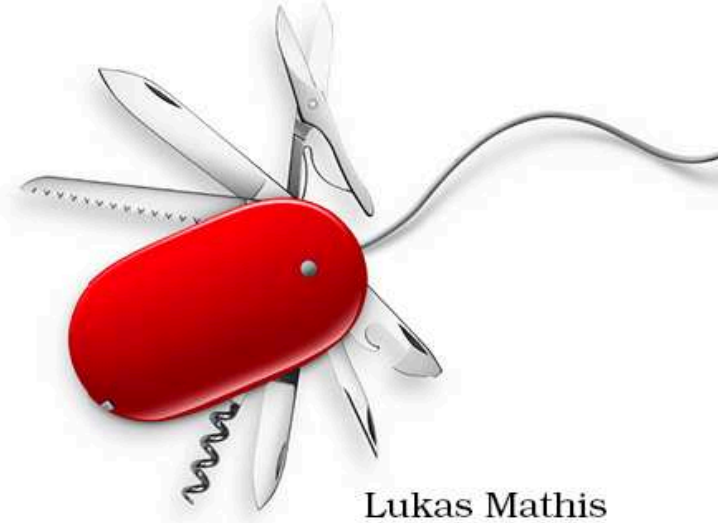
Designed for Use

Usable Interfaces for Applications and the Web

The Pragmatic Programmers

Designed for Use

Create Usable Interfaces for Applications and the Web



Lukas Mathis

edited by Jill Steinberg

Before We Start, a Word

This is a book for visual designers and programmers. It's not, however, about visual design or about code. Instead, it's about something much more important: the people who will be using your product.

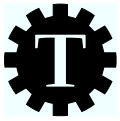
The best product is of no consequence whatsoever if people don't use it. You can create the most beautiful, sturdiest, most elegant brush in the world, but if nobody uses it to paint a picture, your work was in vain.

This book helps you make products—applications and websites—that people will want to use.

There are two kinds of chapters in this book: "technique chapters" and "idea chapters." Each technique chapter explains a specific technique you can use during the design process to make your product more user-friendly: storyboarding, usability tests, or paper prototyping, for example. Technique chapters explain concrete things you can do—the tools for your designer's tool belt.

Idea chapters, on the other hand, talk about ideas or concepts in more general terms: how to write usable text, how realistic your designs should look, when to use animations, and so on. Idea chapters explain things to think about and consider while coming up with designs.

Technique Chapters



You can identify technique chapters by the cog on the first page.

All technique chapters follow the same basic outline. Since not all techniques work well in all situations, I start by quickly outlining the kinds of situations to which the technique applies. Then, I explain what the technique is and how to use it. I end many of the technique chapters with a specific example of the technique as applied to a fictional application we design as we proceed through the book.

Since Twitter[1] apps are our generation's "Hello World" example application, for the technique chapters we'll design a Twitter app. To make things interesting, we're not designing a generic Twitter app. Our app is aimed at people who have to update Twitter accounts for their companies. We call this fictional application BizTwit.

Think of the technique chapters as recipes. It's OK to read the book from start to finish, but it's also OK to delve into a specific topic. To that end, these chapters are typically short and to the point, and they contain references to further information both inside the book as well as in other books or on the Internet.

Idea Chapters

While technique chapters introduce specific techniques and explain how to apply them, idea chapters are less specific. They introduce concepts and are mostly meant as sources of inspiration, rather than as strict rules. Some of the idea chapters mention techniques or refer to technique chapters, but they *focus* on more general concepts: How realistic should design be? How can we use animation most effectively? What are modes? What can we learn from video games?

You can identify idea chapters by the light bulb on the first page.

The ideas in these chapters may not always apply to the projects you're working on, because to some degree, people are unpredictable. When using your products, they don't always behave as you expect them to behave. And they don't always act as your rules predict.

To illustrate how people's behavior is often different than predicted, let's look at an example outside of user interface design. Let's assume you are concerned with public health and safety. Where do you start? Given that tens of thousands of cyclists are injured in traffic accidents every year, bicycle safety is a good place to start.

Studies show that helmets help cyclists avoid injuries. So, getting people to wear helmets should decrease the number of injuries, thereby increasing people's health and safety. The predicted outcome seems

1. In case you don't know what Twitter is (possibly because you're reading this book in the year 2053, when brainjacking is how people communicate), Twitter (at http://twitter.com) is a popular Internet service that people use to publish short text messages—*tweets*—and subscribe to other people's messages.

Typing Web Addresses

This book contains a lot of web addresses. Some of them are pretty long. Maybe you're reading a printed version of this book. Copying these long addresses from your book to a web browser can be cumbersome. To make it a little bit easier, I've set up http://designedforuse.net. This site contains a list of all the long addresses in this book. Instead of typing a long address, type http://designedforuse.net, and click the link there.

obvious: people get into bike accidents, helmets prevent injuries, people who wear bike helmets can avoid injuries. Conclusion: force people to wear helmets.

Over the years, a number of bike-helmet laws have been introduced. However, these laws have not led to the predicted outcome.

In a 2009 study titled "The Health Impact of Mandatory Bicycle Helmet Laws,"[2] Piet de Jong, from the Department of Actuarial Studies at the Macquarie University in Australia, evaluated the effects of such laws. He discovered that people really don't like bike helmets, so much so that many of them simply stop using their bikes altogether if they are forced to wear helmets while riding.

This outcome prompted de Jong to conclude that bike-helmet laws actually have a *negative* effect on societal health as a whole. Yes, the laws prevent some injuries, but for people who stop using their bikes entirely (and often use their cars instead), the health consequences are overwhelmingly negative.

The bottom line is, no one bothered to *test* the laws before enacting them. The people who were affected by the laws did something completely unexpected by the people who designed the laws.

You will often observe the same effect when designing user interfaces. Design changes don't always create the result you intended and sometimes have the opposite effect of what you expected.

When you read the ideas and rules in this book, I want you to keep this in mind. You can do your best to come up with a usable solution; you can follow all the rules and make what seem like obviously

2. You can read the study at http://ssrn.com/abstract=1368064.

usable choices when designing your user interface. But people will still surprise you by finding creative ways of misunderstanding your application's user interface, getting lost on your website, behaving in unpredictable, seemingly illogical ways, and being unable to do the very tasks that seem most obvious to you.

Never assume you can apply a list of usability rules to a product and end up with something usable. Use common sense when designing user interfaces, but don't *rely* on it. Know the rules, but break them if it improves your product. The point is not to do exactly what I tell you to do but instead to take my words as a source of inspiration—and to always test your designs.

How the Book Is Organized

The chapters in this book are presented roughly in the order in which they are applicable during a typical design process, which I've divided into three stages: research, design, and implementation.

Research
: It's tempting to jump right in and start designing a product as early as possible (or perhaps even to start writing code if you're a programmer). In some cases, that may be OK, but it's usually better to start by doing a bit of research. Who is your product for? What problems do you want to solve?

Design
: Think about how to solve your audience's problems. Design solutions and then test them before writing any code. Fixing mistakes on paper is a lot easier than fixing them in code.

  From a design point of view, this stage is probably the most important in the development process and, consequently, represents the largest part of the book.

Implementation
: Create the product, but keep testing it. Were your earlier assumptions correct? Does your design work? How do people interact with it now that it's running? Is your implementation good enough? How does your product deal with errors and real data? Does it perform well enough?

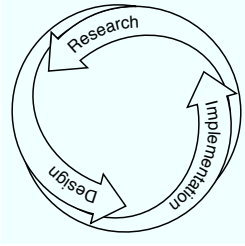
Deciding where to put idea chapters was more of a gut call than an exact science. I've put these chapters where you're likely to find them

useful, but most ideas are applicable most of the time. The organization is more pertinent for technique chapters.

I introduce each technique chapter with a timeline that looks like this:

This timeline should help you understand when a technique is most important or most commonly used. The example timeline indicates a technique that is typically used at the beginning of the "implementation" part of the product development process. However, many techniques are useful at different times of the design process. The timelines are there to help put techniques into context, not as strict rules.

Now, this representation makes it look like the typical development process is a linear affair that goes from research to design to implementation. But typically, design processes are iterative. Your development process is more likely to look a bit like this circle.



However, since we often think of our development process as a number of linear iterations on a product, the linear timeline should be easy to understand.

Just One More Thing

Before we start, I should note that this book has its own web page.[3] It offers a book forum and an errata page. Of course, now that I type this, the errata page is still empty, but by the time you read it, it probably won't be.

And with that out of the way, let's get started!

3. You can find it at http://www.pragprog.com/titles/lmuse.

The Pragmatic Bookshelf

The Pragmatic Bookshelf features books written by developers for developers. The titles continue the well-known Pragmatic Programmer style and continue to garner awards and rave reviews. As development gets more and more difficult, the Pragmatic Programmers will be there with more titles and products to help you stay on top of your game.

Visit Us Online

Homepage for Designed for Use
http://pragprog.com/titles/lmuse
Source code from this book, errata, and other resources. Come give us feedback, too!

Register for Updates
http://pragprog.com/updates
Be notified when updates and new books become available.

Join the Community
http://pragprog.com/community
Read our weblogs, join our online discussions, participate in our mailing list, interact with our wiki, and benefit from the experience of other Pragmatic Programmers.

New and Noteworthy
http://pragprog.com/news
Check out the latest pragmatic developments, new titles and other offerings.

Buy the Book

If you liked this eBook, perhaps you'd like to have a paper copy of the book. It's available for purchase at our store: pragprog.com/titles/lmuse.

Contact Us

Online Orders:	www.pragprog.com/catalog
Customer Service:	support@pragprog.com
Non-English Versions:	translations@pragprog.com
Pragmatic Teaching:	academic@pragprog.com
Author Proposals:	proposals@pragprog.com
Contact us:	1-800-699-PROG (+1 919 847 3884)