Painless Project Management with FogBugz

Second Edition

Mike Gunderloy

Apress®

Painless Project Management with FogBugz, Second Edition

ISBN-13 (pbk): 978-1-59059-914-3

ISBN-10 (pbk): 1-59059-914-4

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Lead Editor: Jeffrey Pepper
Technical Reviewer: Joel Spolsky
Editorial Board: Steve Anglin, Ewan Buckingham, Gary Cornell, Jonathan Gennick, Jason Gilmore, Jonathan Hassell, Chris Mills, Matthew Moodie, Jeffrey Pepper, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh
Senior Project Manager: Beth Christmas
Copy Edit Manager: Nicole Flores
Senior Copy Editor: Ami Knox
Assistant Production Director: Kari Brooks-Copony
Senior Production Editor: Laura Cheu
Compositor: Jimmie Young/Tolman Creek Media
Proofreader: Linda Seifert
Indexer: John Collin
Artist: Kinetic Publishing Services, LLC
Cover Designer: Kurt Krames
Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail orders-ny@springer-sbm.com, or visit http://www.springeronline.com.

For information on translations, please contact Apress directly at 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit http://www.apress.com.

For online friends everywhere, but especially in Second Mirage

Contents at a Glance

Contents

And so, on the Upper West Side of Manhattan, if you're a restaurant, and you want to thrive, you have to carefully debug everything.

You have to make sure that there's always someone waiting to greet guests. This person must learn never to leave the maitre d' desk to show someone to their table, because otherwise the next person will come in and there will be nobody there to greet them. Instead, someone else needs to show patrons to their tables. And give them their menus, right away. And take their coats and drink orders.

You have to figure out who your best customers are—the locals who come on weekday nights when the restaurant is relatively quiet—and give them tables quickly on Friday night, even if the out-of-towners have to wait a little longer.

You need a procedure so that every water glass is always full.

Every time somebody is unhappy, that's a bug. Write it down. Figure out what you're going to do about it. Add it to the training manual. Never make the same mistake twice.

Eventually, Isabella's became a fabulously profitable and successful restaurant, not because of its food, but because it was debugged. Just getting what we programmers call "the edge cases" right was sufficient to keep people coming back, and telling their friends, and that's enough to overcome a review where the *New York Times* calls your food "not very good."

Great products are great because they're deeply debugged. Restaurants, software, it's all the same.

Great software doesn't crash when you do weird, rare things, because everybody does something weird.

Microsoft developer Larry Osterman, working on DOS 4, once thought he had found a rare bug. "But if that were the case," he told DOS architect Gordon Letwin, "it'd take a one in a million chance for it to happen."

Letwin's reply? "In our business, one in a million is next Tuesday."[2]

Great software helps you out when you misunderstand it. If you try to drag a file to a button in the taskbar, Windows pops up a message that says, essentially, "You can't do that!" but then it goes on to tell you how you can accomplish what you're obviously trying to do. (Try it!)

Great software pops up messages that show that the designers have thought about the problem you're working on, probably more than you have. In FogBugz, for example, if you try to reply to an e-mail message, but someone else tries to reply to that same e-mail at the same time, you get a warning and your response is not sent until you can check out what's going on.

Great software works the way everybody expects it to. I'm probably one of the few people left who still closes windows by double-clicking in the top-left corner instead of clicking the X button. I don't know why I do that, but it always works, with great software. Some software that I have is not so great. It doesn't close if you double-click in the top-left corner. That makes me a little bit frustrated. It probably made a lot of people frustrated, and a lot of those people probably complained, but I'll bet you that the software developers just didn't do bug tracking, because they have never fixed that bug and probably never will.

---

2. Osterman, Larry. "One in a million is next Tuesday," from Larry Osterman's WebLog (personal web page). `http://blogs.msdn.com/larryosterman/archive/2004/03/30/104165.aspx`, dated March 30, 2004, retrieved December 9, 2004.

What great software has in common is being deeply debugged, and the only way to get software that's deeply debugged is to keep track of your bugs.

A bug-tracking database is not just a memory aid or a scheduling tool. It doesn't make it easier to produce great software, it makes it possible to create great software.

With bug tracking, every idea gets into the system. Every flaw gets into the system. Every tester's possible misinterpretation of the user interface gets into the system. Every possible improvement that anybody thinks about gets into the system.

Bug-tracking software captures the cosmic rays that cause the genetic mutations that make your software evolve into something superior.

And as you constantly evaluate, reprioritize, triage, punt, and assign these flaws, the software evolves. It gets better and better. It learns to deal with more and more weird situations, more and more misunderstanding users, and more and more scenarios.

That's when something magical happens, and your software becomes better than just the sum of its features. Suddenly it becomes reliable. Reliable, meaning it never screws up. It never makes its users angry. It never makes its customers wish they had purchased something else.

And that magic is the key to success. In restaurants as in software.

Joel Spolsky

About the Author

MIKE GUNDERLOY has been involved with the computer industry for over a quarter century. In that time, he's assembled PCs, run network cable through drop ceilings, contributed to and managed software projects large and small, as well as written many books and articles. When he's not banging out code or words on the keyboard, Mike raises children, turkeys, and tomatoes on a small farm in eastern Washington State.

About the Technical Reviewer

JOEL SPOLSKY is an expert on software development and the founder of Fog Creek Software. His Web site, Joel on Software (`http://www.joelonsoftware.com`), is popular with software developers around the world and has been translated into over 30 languages. His latest book is *Smart and Gets Things Done: Joel Spolsky's Concise Guide to Finding the Best Technical Talent* (Apress, 2007).

Acknowledgments

Every book starts somewhere. In the case of the current volume, the starting point is easy to pinpoint: an e-mail from Fog Creek Software's Joel Spolsky, asking me if I'd be interested in putting together a book about FogBugz 4.0. After I leapt at (I mean, after I carefully considered and accepted) this proposal, Gary Cornell at Apress was instrumental in pulling together the contractual details necessary to make this book a reality. When Joel came back to me asking for a revised version of the book to cover FogBugz 6.0, I was happy to leap, er, consider and accept once again.

Of course, a book about software can't be written without the software itself, and in this case it's easy to know who to thank for that: Joel and the rest of the Fog Creek staff. Special thanks to Michael Pryor, who spent a few maddening hours logged in to one of my computers remotely trying to figure out what I was doing to provoke a particularly obscure bug. Beyond that, the active FogBugz user community is to thank for many of the innovations in this version of FogBugz.

I'd like to thank Project Manager Beth Christmas, Copy Editor Ami Knox, and Production Editor Laura Cheu for their hard work turning my manuscript into something actually resembling a book. And let's not forget the hard-working production crew: Compositor Jimmie Young, Proofreader Linda Seifert, and Indexer John Collin.

Then there are the people outside of the actual book production process who still deserve huge thanks: my family. So thanks and much love to my dear wife, Dana, and our kids, Adam, Kayla, Thomas, and Lindsey.

Introduction

Like any other developer who wants to actually ship software, I use software management tools. One of the most important tools in my own toolbox is FogBugz. Now on its sixth major release, FogBugz is a complete project management system optimized for software teams. It's Web-based, so you access most of the functionality through your Web browser. I've found this an immense help when working with developers and testers scattered about the Internet, but you can also use FogBugz for projects that are maintained entirely at a single location. In this book, you'll see why I'm so excited about FogBugz, and learn what it can do for your own software management tasks.

Why FogBugz?

Many of the software applications that overlap the functionality of FogBugz present themselves as bug-tracking systems, but there's more to FogBugz than just tracking bugs. FogBugz is a tool for tracking, updating, and managing cases. There are four kinds of cases:

- *Bugs*: Things that don't work right
- *Features*: New things being planned
- *Inquiries*: Questions from customers or team members
- *Schedule items*: Large chunks of time such as integration that have a critical impact on ship dates.

Every case is prioritized, categorized, and assigned to exactly one person on your team who must either resolve it or assign it to someone else. Developers work through their cases one by one, ideally in order of priority. That doesn't sound like much to handle, but FogBugz integrates case tracking with many other features, including the following:

- Source code control integration, which makes it easy to see which check-ins are associated with which bugs or features, and allows you to set up an elegant online code review system.
- Filters and advanced full-text search that make it easy to sort and search.
- A built-in estimation system that learns from the past to help you track your project and ship on schedule.

- Automatic release note generation from the cases that were resolved for a particular release.
- A customer e-mail management facility that discards spam and sorts the mail into categories based on your own training. FogBugz preserves the entire e-mail history and makes it easy to keep the customer informed of progress on a case.
- A wiki that provides collaborative editing for everything from requirements documents to weekly status reports.
- Integrated discussion groups for customers, testers, or team members. Discussion groups include anti-spam features and easy integration with case tracking.

What's in This Book

My goal in this book is to take you from the very basics of FogBugz through all the details of managing and administering a complex FogBugz installation. Depending on your role—developer, tester, manager, system administrator, or (as with many of us) jack-of-all-trades—you may want to read some portions of the book more closely than others. Here's a roadmap of what you'll find inside:

In Chapter 1, you'll learn about the overall philosophy of FogBugz (yes, software applications have philosophies) and get an introduction to how FogBugz works in practice. I'll take you through the life cycle of several cases so that you can get a feel for how the pieces fit together.

Chapter 2 concentrates on the actual process of case management with FogBugz. You'll learn how cases get into the system and how to deal with them once they've been entered. This chapter covers taking screenshots and attaching files, as well as filtering and sorting to find the cases that you need.

Chapter 3 is directed mainly at the FogBugz administrator. While many organizations will be able to use FogBugz productively right out of the box, there are quite a few pieces of the program that you can customize. If you're the one responsible for fine-tuning FogBugz where you work, this is the chapter for you. You'll learn how to set up projects, areas, clients, departments, and much more.

Chapter 4 looks at FogBugz from the perspective of the software manager. This is the chapter that covers estimating techniques, due dates, the proper way to resolve cases, and release notes. It also introduces the innovative evidence-based scheduling (EBS) system that allows FogBugz to generate supremely accurate estimates of project ship dates.

Chapter 5 covers customer and team communication via wiki, e-mail, and discussion groups. These features let you connect your FogBugz database directly with your team and your customers, tapping their collective intelligence and excitement.

Finally, in Chapter 6, I cover the integration of FogBugz with other tools, including your source code control system, IDE, and reporting tools. You'll learn about the various extensibility points that make FogBugz an open system.

The book wraps up with an appendix that reviews the instructions for installing FogBugz on Windows, Linux, or Mac OS X servers.

I hope that by the end of the book you'll consider yourself a serious FogBugz user, and that you'll find this program as useful and well designed as I do.

Contacting the Author

I'm always happy to hear from readers of any of my books. You can find my own Web site at `http://www.larkware.com`, or e-mail me directly at `MikeG1@larkfarm.com`. But in the case of FogBugz questions, there's another resource you should try for a quick response: the FogBugz discussion group at `http://support.fogcreek.com/?fogbugz`. You'll find many passionate and committed FogBugz users there who are happy to help out, as well as Fog Creek's own support staff.