Extracted from:

Explore It!

Reduce Risk and Increase Confidence with Exploratory Testing

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina

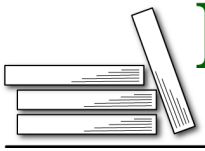
The Pragmatic Programmers

Explore It!

Reduce Risk and Increase Confidence with Exploratory Testing

Elisabeth Hendrickson

Edited by Jacquelyn Carter



Pragmatic Bookshelf

Our Pragmatic courses, workshops, and other products can help you and your team create better software and have more fun. For more information, as well as the latest Pragmatic titles, please visit us at *http://pragprog.com*.

The team that produced this book includes:

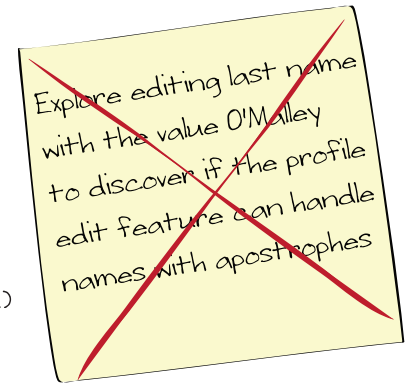
Jackie Carter (editor)
Potomac Indexing, LLC (indexer)
Molly McBeath (copyeditor)
David J Kelly (typesetter)
Janet Furlow (producer)
Juliet Benda (rights)
Ellie Callahan (support)

Printed in the United States of America.
ISBN-13: 978-1-937785-02-4
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—February 2013

2.3 Good Charters

A good charter offers direction without overspecifying test actions. As an example, the following isn't a charter; it's a test case.

Too Specific!
(There isn't much to explore. Plus, it's like a weirdly worded test case.)

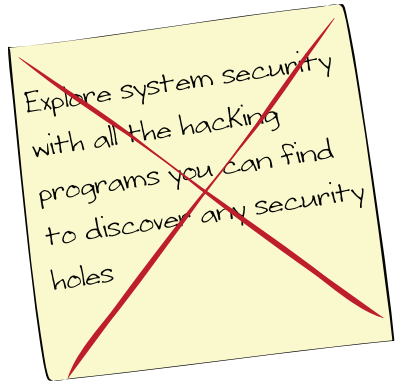


When charters are too specific, they become just a different (and weird) way of expressing individual tests. We end up spending a lot of time on test documentation with very little benefit.

On the flip side, charters that are too broad run the risk of not providing enough focus. You won't know how to tell when you're done exploring if the target is too big.

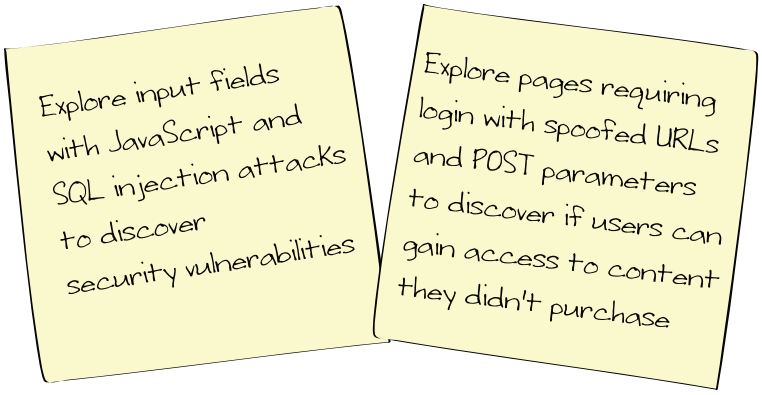
For example, consider this charter:

Too Broad!
(You can never explore enough to fulfill this mission.)



It's so vague that you would never finish the mission. It calls for exploring the entire system with a large and undefined set of resources. You could spend weeks investigating it and still not be sure you discovered all the important risks and vulnerabilities.

Rather than a single huge charter, it is better to craft multiple charters, where each focuses on a single area and/or a specific type of security hole:



A good charter is a prompt: it suggests sources of inspiration without dictating precise actions or outcomes.

2.4 Generating Charters

Your explorations yield information, but unless your stakeholders value that information and use it to move the project forward, you're wasting your time. To ensure the information you find has value, you need to work closely with your stakeholders to identify and frame charters targeted at answering the most valuable questions about the software.

This section examines some of the sources that inspire your charters.

Requirements

Requirements discussions are an ideal time to start drafting charters. Let's see how this works by eavesdropping on a conversation between Alex, a tester; Pat, a programmer; and Binh, a business analyst. They're discussing the feature that allows users to update their profile information.

PAT: *So, which of the profile fields should be modifiable?*

BINH: *All of them.*

ALEX: *Even the username? So someone with the username "fred728" can change his username to "iamfred" and then use the new username to log in?*

BINH: *Yup!*

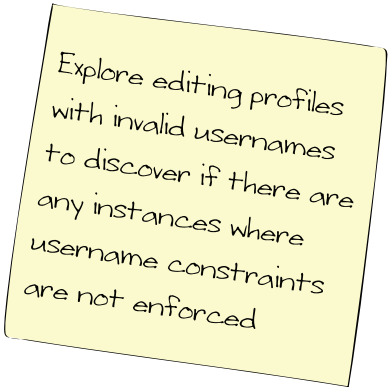
ALEX: *Wow. I'm concerned about the possibility of violating the restrictions around usernames if we allow users to change their usernames after the fact.*

PAT:

That won't be a problem. We've isolated all the username constraint logic to a single place in the code base. It will behave exactly the same as if a user is creating an account.

ALEX: *That sounds like something I should explore.*

Pat is asserting that there won't be any problems, but Alex has seen plenty of cases where the programmer was surprised by the software's behavior. He knows better than to assume Pat has the whole code base memorized. Alex makes a note:



The conversation continues. Alex and Pat have questions about the interactions between this feature and other capabilities that have already been implemented:

PAT: *Should users be able to update their profiles if their accounts are suspended?*

BINH: *Oooh. Good question. I'll have to think about that.*

ALEX: *I bet there will be interactions between updating profiles and account states. I'd better explore around that.*

Aha! Binh might come back with a simple yes or no answer to this specific question, but there is a deeper issue at play here. The first clue is that Binh says, "I'll have to think about that." The second clue is that there is a potential for an interaction between the new feature and existing capabilities.

Any time a question reveals uncertainty, ambiguity, or dependencies, there's something important to be explored during development. So Alex captures a charter:

As the discussion continues, Alex hits on an idea that wasn't on Binh's radar:

> ***ALEX:*** *Users see their profile information on their account page and also on the dashboard. So far we're talking about editing their profiles from the account page. Should we also allow users to update their account information from the dashboard?*
>
> ***BINH:*** *I think that's out of scope for now, but it's something we should consider in the next update. I'll add it to my list.*

Exploring can reveal opportunities to add new requirements as well as find risks or problems. When you begin questioning and exploring, you can watch for such opportunities, then review your suggestions with your stakeholders.

Implicit Expectations

In this example, Pat and Alex are asking Binh about his expectations. However, no matter how much Pat and Alex probe, Binh will still have additional expectations that he doesn't even think about expressing. Binh might think that a given expectation is too obvious to mention. Here's what one product manager said to me when I asked about the interaction between a new feature and the existing security features: "We have a security model in this system. New features have to honor that security model. I just expect you guys to take that into account without my saying it for every single feature."

Another example of implicit expectations might include crosscutting quality criteria such as reliability, scalability, or performance. If the functionality works as specified but the response time increases from under a second to over a minute, there's a problem, even if the response time wasn't explicitly stated as part of the requirement.

Whenever you recognize an implicit expectation that deserves exploration, capture it as a charter.

Charters Align Goals

The requirements discussion is an ideal time to get feedback on the extent to which your ideas about the most important risks match the rest of your stakeholders' concerns. As you think of possible charters, you can ask your stakeholders if they would value the information those charters might reveal. Pat, Alex, and Binh might consider questions like these:

- "Should we look for possible performance implications?"
- "If there were problems with legacy data, would we want to know?"
- "If we could find a way that users could put their accounts into an unusable state, no matter how crazy, we'd fix it, right?"

Asking these questions is important because you don't want to spend a lot of time discovering information that no one will ever take action on. It's a waste of time. For example, Binh might say, "If users do something really crazy to their accounts, the Help Desk can get them sorted out. Update Profile should use the same validations as Create Account, but nothing beyond that." This would imply that it's worth exploring updating the username, but it isn't worth spending days exploring conditions that go beyond the basic validation that Pat says is already built in.

Of course, you might disagree with your stakeholders about risk. The best time to surface that disagreement is during a discussion like this, before you have spent hours exploring. Perhaps you see a particular kind of risk that the rest of your stakeholders don't see. Discussing the risk in advance can either allay your concerns or raise your stakeholders' awareness.

Stakeholder Questions

Questions surface throughout the development cycle. Indeed, sometimes the very best questions come up when mulling the implications of a given design decision or when mapping out a connected set of features. For example, if you were working on software that had both privacy settings and a messaging feature, your stakeholders might become concerned about their interaction:

> How do the existing privacy settings interact with our new messaging feature?

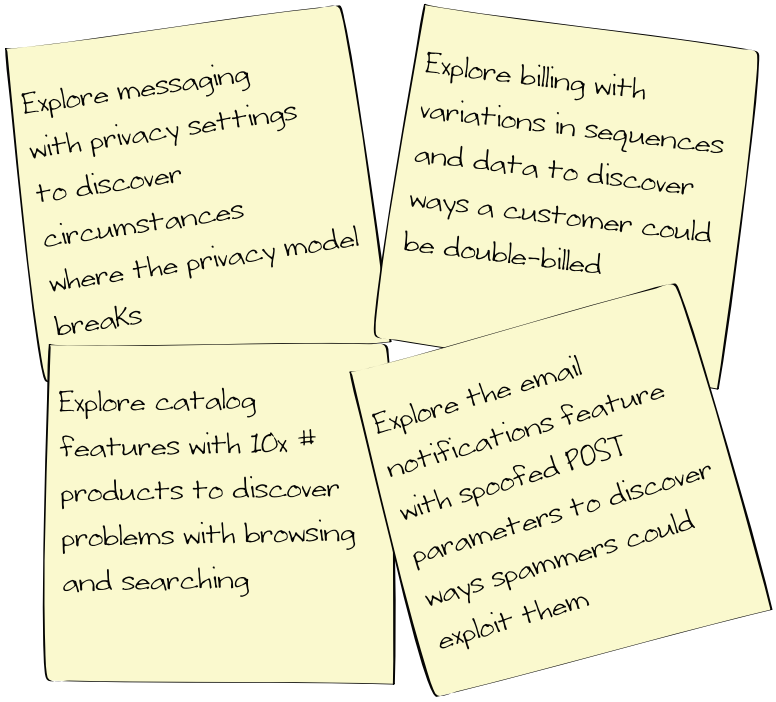
Your stakeholders might wonder what will happen in the future:

> What will happen if we add ten times the number of items to the catalog?

Your stakeholders might become worried about possible risks:

> Could a hacker hijack the email notification system to send spam?
>
> Are there any circumstances under which a customer could be double-billed?

These questions can form the basis for charters like these:



Existing Artifacts

The source code can even yield interesting charter ideas, particularly if you happen across a code comment like this one:

```
// I don't know why this works, but it does. Don't touch it.
```

Other existing artifacts associated with the software can yield chartering ideas. The bug database is likely to offer up a wealth of insight about historic areas of risk. Scanning the logs from support calls can give you insight into the risks that have historically bitten customers.

New Realizations and Discoveries

Chartering is an ongoing process. You start chartering as soon as anyone starts discussing requirements, and you continue identifying charters throughout development. As you explore, it's also normal for you to realize that the charters you have mapped out only barely scratched the surface.

You can tell this is happening when you start executing sessions against a charter only to discover that you're continually tempted to explore in directions

that are decidedly off-charter and you're afraid to ignore these temptations for fear that you won't remember to come back.

Such temptations are a cue that you need to jot down additional charters to pursue in later sessions.