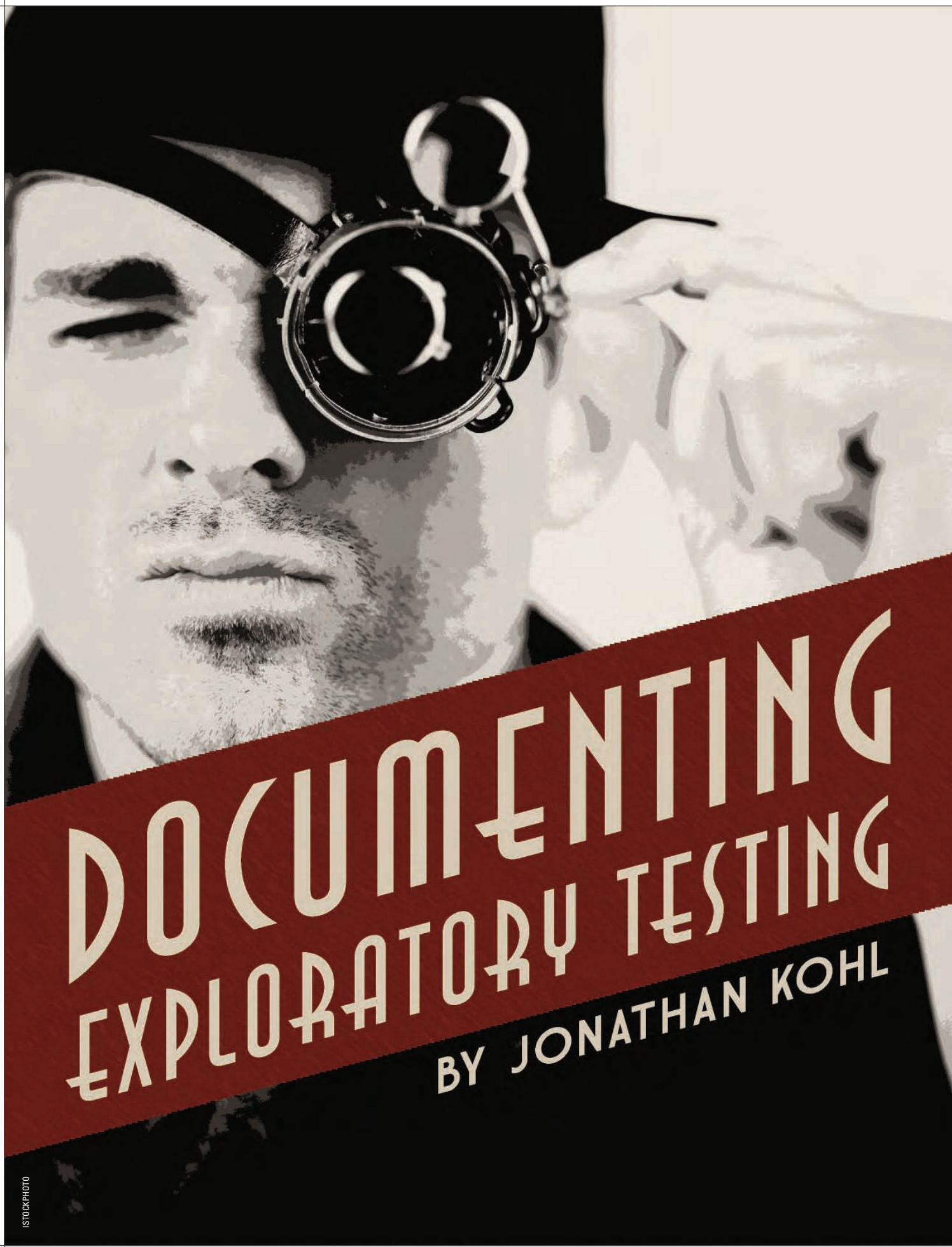

# DOCUMENTING EXPLORATORY TESTING

BY JONATHAN KOHL

ISTOCKPHOTO

JUST BECAUSE WE ARE USING AN EXPLORATORY TESTING APPROACH ON OUR PROJECTS DOESN'T MEAN THAT WE DON'T DOCUMENT OUR WORK. With exploratory testing, you can document as much or as little as required by your stakeholders.

Determining documentation requirements on software projects isn't difficult; it just takes a bit of investigation. For example, while working on a project in a regulated environment, I talked to the people in charge of regulatory issues for the company and asked them what documentation they needed. They said they needed a risk assessment, a test plan, test cases, and test results. I had everything but test cases, so I pressed further. They wanted test cases simply because that's what they were used to. The regulatory requirements didn't ask specifically for test case documents; there just needed to be enough documentation so that tests could be repeated by others.

## Planning and Strategy

### WHAT DOES A TEST PLAN LOOK LIKE IN EXPLORATORY TESTING?

Often, test plans that direct exploratory testing look very similar to their scripted testing counterparts. On any project I am leading, I go through analysis and planning to determine how to optimize the use of our people, tools, equipment, budgets, etc. A risk assessment (required by auditors) can help focus our testing.

To create a risk assessment, I do some research. I talk to marketing, sales, and product management and ask them about core functions of the product. What is the purpose of the software? What kinds of tasks would users expect to perform with the software? What would product success or failure look like?

Next, I talk to the technical team. What features are new? Are there any technology changes that are required, such as new tools, libraries, or hardware? Have they discovered any challenging areas in their work? How do they define success and failure?

I then talk to product management about quality criteria. What characteristics are they expecting from the software? What about technical areas, such as security, accessibility, and performance?

The risk assessment can be a list of items describing how we would mitigate those risks through testing, a focused testing approach for a specific risk, or a recommended technique, such as performance or security testing. Figure 1 shows an example of a risk assessment.

After developing the risk assessment, I look at a test strategy. What is the purpose of our testing? Sometimes, project stakeholders have a clear objective for testing; other times, it isn't as clear. I provide options to stakeholders. "Do you want us to find important bugs as quickly as possible, or

| Session Tester 0.3 Release Risk Issue | Potential Consequence or Outcome | Outcome Tolerable? | Probability of Occurrence | Mitigation Approach by Testing |
|---|---|---|---|---|
| ***Business Related*** | | | | |
| New workflow: application loads in full screen mode, not session-start mode. | Usability may be impacted negatively. The application may load slowly. | No | Medium | *Scenario testing on all supported platforms, focused exploratory testing* |
| Text editor changes | The notes that are entered may not be saved accurately. There may be a performance impact since we are using a 3rd party library instead of a lightweight parser. | No | High | *Regression test simple sessions. Focused ET using test data generation to try different types of data, different sizes of data, and each markup tag,* |
| Project workflow is new | The application may not work properly. It might stop responding, crash or behave strangely. Users may not understand the workflow. | No | Low | *Functional Testing, User Scenario Testing, Usability testing. Focused ET in error generation workflows.* |
| View/edit sessions is new | Time-sensitive information may get corrupted in edit mode. | No | High | " |
| | | | | |
| ***Technology Related*** | | | | |
| New XQuery library for reporting | Reporting information might get lost or corrupted. | No | Medium | *Testing using generated XML session docs using different charactersets, time-related data.* |
| New SDK used for development | Some platforms (notably Mac) may not support all the features. | No | Medium | *Functional and Regression testing on each supported platform* |
| Using syntax library as text editor | We may use the API improperly, causing unintended consequences for end users. | No | Medium | *Requirements-based testing, user scenario testing, data generation using a tool.* |

**Figure 1: Sample risk assessment**

are you more concerned that our testing mitigates the risks outlined in the risk assessment?" Once I have determined a purpose for our testing work, I can look at who is available and start to outline what tools, practices, and tactics we will use when testing.

Next, I look at project resources. Which people are available and when? What tools, machines, software, and other resources do I need to complete this project? Once I have clear answers to these questions, I start to glue together my project analysis, risk assessment, strategy thoughts, and logistics. I learned this from James Bach [1]. If this needs to be formalized in a document, this becomes the basis for my test plan.

Traditional testers love test plans, but many exploratory testers hate them. I have a secret: I need to do all of that work, anyway, so if I document bits of it as I go, taking that existing information and plopping it in a test plan document is simple and fast. If I need to change the test plan into a different format or adapt because the project changes mid-course, it's easy and quick.

## Guidance Documents

### How do new people learn without test cases?

Test cases aren't necessarily the best way to learn how to test new software. Alternatively, I use tools and techniques for learning that are proven in training and documentation disciplines. For example, in the regulated-environment project discussed earlier, instead of relying on test cases, we created a "guidance" folder on a shared network drive. I found existing sources of documentation from the marketing team (sales brochures, demo presentation materials), the documentation team (in-progress user manuals), and the operations team (installation and administration guides). That gave us a start but didn't provide enough information for testing. Using documentation department templates, we started writing "how to" documentation with a testing focus. We used screen-recording software to record demo videos since they were easier to maintain than written documentation. If specific setup was needed, we created "calibration" documents that provided step-by-step instructions. To help with test idea generation, we created testing cheat sheets with different testing ideas.

To help focus testing in specific areas, guidance can be provided in the form of checklists. Each coverage model can generate checklists with corresponding guidance information in written and video form.

Checklists essentially are lists of test ideas. The lists are specific enough to provide focus but vague enough to allow for personal interpretation and freedom to explore. Figure 2 shows a sample checklist.

This is how a tester would use documentation to help

**Session Tester 0.3 Coverage Outline**

| Revision: | r242 | | | |
|---|---|---|---|---|
| | | | | |
| **Application Load** | | | | |
| Full application defaults | pass | | | |
| Treeview | pass | | | |
| Session buttons other than "Start" disabled | pass | | | |
| | | | | |
| **Project** | | | | |
| New Project wizard | | | | |
| Project Tree View actions | | | | |
| Edit Project | pass | | | |
| New project with tags | | | | |
| | | | | |
| **New Session** | | | | |
| Session buttons enabled | pass | | | |
| Timer | pass | | | |
| Date/Time | pass | | | |
| Mission | pass | | | |
| Primer | pass | | | |
| Project relationship | fail | | | |
| | | | | |
| **Session State** | | | | |
| Text Editor | | | | |
| Timer | pass | | | |
| Autosave | | | | |
| Tags | | | | |
| Pause | | | | |
| Stop | | | | |
| View session | | | | |
| | | | | |
| **Reports** | | | | |
| Html Reports | | | | |
| Structure | fail | | | |

**Figure 2: Sample test checklist excerpt**

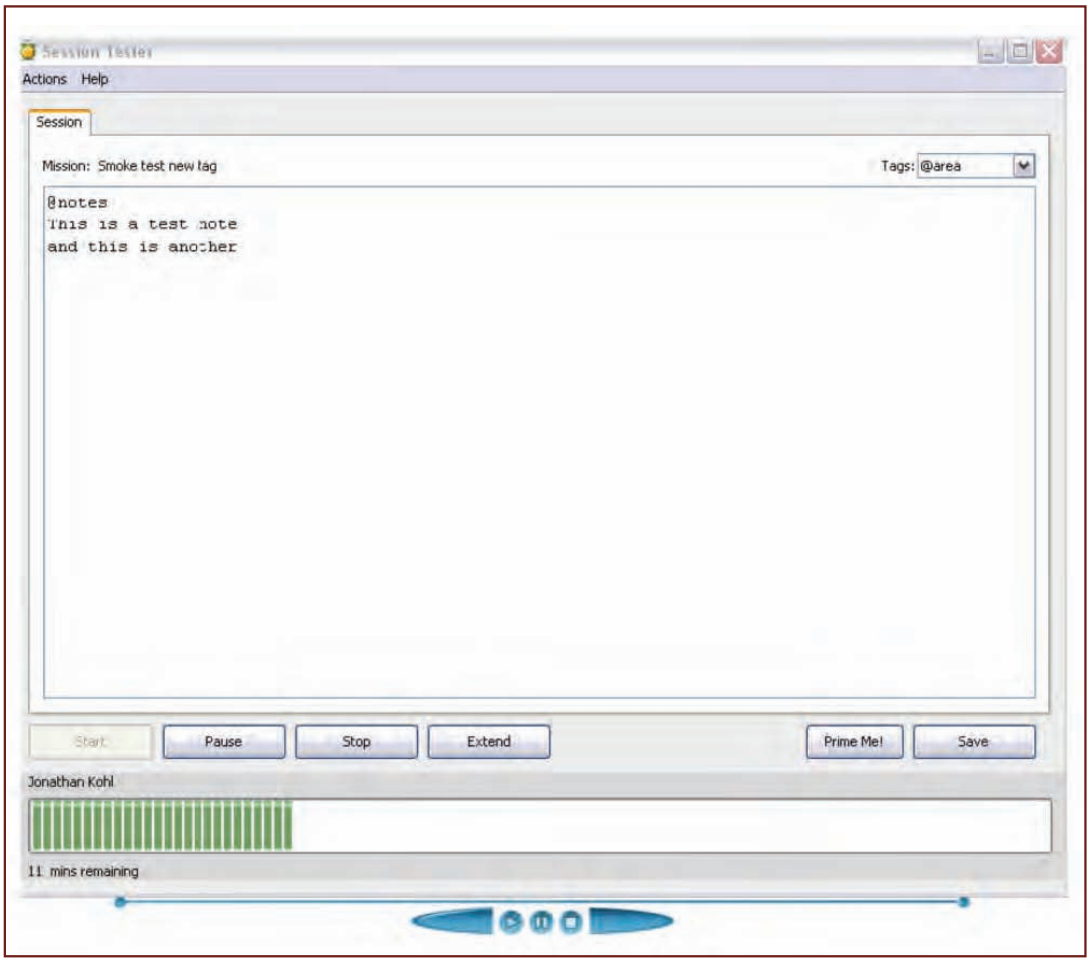


**Figure 3: A test video screen capture**

guide his exploratory testing: He would look up a specific test checklist related to the area of the program he was testing. If he was unfamiliar with that application or feature set, he would read a how-to guide document to learn how to use the software and watch videos that demonstrate how to use and test the software. Next, he would start testing the items listed in the checklist, referring to the guidance information as needed. When he needed more test ideas, he'd review testing cheat sheets to help trigger more creative ideas.

## Documenting Test Execution

### How can we demonstrate what was tested?

Session-based test management is a high-accountability approach to exploratory testing invented by James and Jon Bach. This is a descriptive approach to testing, where you document what you are testing as you go. The traditional test case model is prescriptive—you document in advance what you would like tested. Both approaches generate documentation that can be reviewed and audited. I use lightweight implementations of session-based test management. Session ideas focus on specific testing tasks. Testing sessions are uninterrupted periods of testing time lasting from sixty to 120 minutes. After an exploratory session, testers review their session findings with colleagues. This helps generate follow-on test ideas and is great for knowledge transfer. If there are regulatory or auditing requirements, a session sheet can be used as a project artifact.

If traceability is needed, session sheets can map to coverage outlines and to guidance documents and media. For example, an auditor could ask about a specific bug that was logged and be referred to the session where the bug was discovered. From the session sheet, the auditor could review the coverage outline checklist and the guidance documentation. Watching a short video of the bug report and contrasting it with the how-to testing document and video, he could repeat the test himself and verify that the bug had been fixed. See figure 3 for an example.

While the auditors on our example project were at first a bit concerned with a lack of test cases, they loved the use of media and the "how-to" document approach. They found it easier not only to get a sense of what was tested but also to repeat tests themselves. The video was incredibly useful for them, and they liked that session sheets described what actually happened instead of test cases that describe what we expected to happen.

Documenting testing using video has become more popular. It's an inexpensive and powerful way to show people what you mean rather than trying to describe concepts in a document. Recording technology is cheap and readily avail-

able. There are also free software options on a variety of platforms. One of my colleagues recorded, produced, and distributed a testing training video of me demonstrating test techniques—all on his smart phone.

## Documenting Results

### How do we track results and report on test coverage?

On a project that uses free-form exploratory testing without documentation, it can be difficult to explain what was tested. There are lightweight tools that can be used on exploratory testing projects to demonstrate results and progress.

I like to have visible results that show quality and test coverage in simple terms. From there, more detail can be requested and easily supplied without providing much burden on the team. James Bach has a "low-tech" testing dashboard template that can easily be transcribed on a whiteboard. Any team member walking past the whiteboard can get a sense of current quality and testing coverage on the project. From there, checklists can show more detail and session sheets can show actual testing results. This process can easily fit with project management, fault tracking, and other team productivity systems.

I prefer a "pay-as-you-go" approach to documentation. I don't do a lot of speculative, up-front documentation if I don't have to. I want to avoid having to change it frequently as the project adapts and changes. I do a bit of documentation at a time and build toward final products as the project itself evolves. If you are using an agile or iterative lifecycle, you may not need to document all the time, particularly if you start testing at the beginning. If you are in a regulated environment, auditors are used to a "testing phase," so you may be able to use an iteration near the end of the project to create required documentation.

## Conclusion

You can use as much or as little testing documentation as you need on an exploratory testing project. However, make sure that what you do is compatible with the lightweight, test-execution focus of exploratory testing. Do not impose documentation overhead that dominates testing activities—those should focus on testing first. In many cases, you can use something lightweight to solve a problem that seems to require a heavyweight approach. With a little creativity, you can fit your documentation requirements within an exploratory testing approach. **{end}**

jonathan@kohl.ca

**Sticky Notes**

For more on the following topics, go to www.StickyMinds.com/bettersoftware.

- References
- Further reading