This presentation is about how to design tests in an exploratory fashion; aiming to learn more, without stifling execution freedom.

It contains a sub-set of test design heuristics, my current favorites.

Version 1.0
This presentation is about designing tests for this reality, where we don’t know exactly what we are looking for.

“Exploratory software testing is a style of software testing that emphasizes the personal freedom and responsibility of the individual tester to continually optimize the value of her work by treating test-related learning, test design, test execution, and test result interpretation as mutually supportive activities that run in parallel throughout the project.”

[Cem Kaner]

Why is this good?

at start we don’t know everything important
we want to know more
we will design tests and learn as we go
The software potato:
The square symbolizes the features and bugs you will find with test cases stemming from requirements (that can’t and shouldn’t be complete)
The blue area is every possible usage, including things that maybe no customers would consider a problem.
The brown area is what is important, there lies those problems you’d want to find and fix.

This problem has been solved many times at many places with many different approaches. What is common could be that testers learn a lot of things from many different sources, combine things, look at many places, think critically and design tests (in advance or on-the-fly) that will cover the important areas.
Some part luck, and a large portion of hard work is needed. Serendipity is working to our advantage.

Recently I realized there can be more than one potato, that are three-dimensional and perhaps slippery; there might be small potatoes that are the best of them’ all...
In medias res...

You design tests all the time. These activities are sometimes separated, but often intertwined.
Test Execution

- Benevolent start
- Be ready for serendipity
- Variations
- Do one more thing
- Background complexity
- Method acting
- Fresh eyes find failure
- Follow the scent
- Dogfooding

Start nice and easy with big chunks, so you find major, major problems at once. Try to do what is intended, before intentionally trying to break it.

At all times, be ready to find valuable things you didn’t know you were looking for.

Make Variations all the time, broaden your usage with different data, different paths, so you’ll see and learn more.

Whatever test you do, add one more thing, something fast, popular, or error-prone. You might find something important, or learn something.

If you use unnecessarily complex data, you might stumble on important issue. Don’t worry that it will be difficult to pinpoint issues, that’s better than not knowing about them at all.

Try to be a user, try to mimic their needs, feelings, data and environments.

Look at new things, let others look at your things: fresh eyes find failure

When you notice something odd or strange, investigate at once, or let it rest until you get more clues.

If you can use the software for real, you’ll get a better understanding of what and how it should behave. If you’re confident of your product, call this “sip our own champagne”.

Quicktests

- Continuous Use – leave things over weekend
- Interference – remove, rename, cancel, swap, null
- Data Relationships
- Error-Prone Machine
- Coverage For Free
- Use checklists, your own or
  - Hendrickson, Test Heuristics Cheat Sheet
  - Hunter, You Are Not Done Yet
  - Bach, SFDPOT, HICCUPPSF, CIDTESTD, FDSFSCURA

Diverse Half-Measures – “it’s better to do more different kinds of testing to a pretty good level, than to do one or two kinds of testing perfectly.” Lesson 283 i Cem Kaner, James Bach, Bret Pettichord, Lessons Learned in Software Testing

Ongoing test ideas are used by most manual testers, but seldom talked about. For example, you’ll notice and communicate if a dialog takes 15 seconds to display. (Performance) You’ll get annoyed if something is overly cumbersome to do (Usability, Operability) You can improve this skill by learning more about Quality Characteristics, and which matters in your context.
Quality attributes are important for all products, not all of them, but many of them.

Capability is the functionality, probably covered in other ways; but interoperabilities are difficult. Reliability is about handling many situations, and recover from errors. Usability is also the efficiency you as a tester want to have the 100th time you’re using the product. Light Accessibility Testing is very cheap. Charisma is important, but seldom tested; does the product have “it”? Security might seem difficult, but some basics for your specific product you already know. Performance is both on large-scale, but also for each function and environment. Installability, Upgrade and Uninstallation; but also that customer IT department can support and maintain the system and its artefacts. Compatibility with hardware, operating system, software, previous versions, and standards.
Quality characteristics describe attributes that most software benefit from. They can be used on the whole product or for details.

The whole is made by the details. The quality of a detail is defined by the whole.

This is a thorough extension in the same spirit as Bach’s CRUSSPIC STMPL.

Test Analysis

- It’s about finding out what is important
- Exploratory Test Design looks anywhere

- Mary Had a Little Lamb Heuristic

- Change granularity

- Question everything, especially this statement

- There are many information sources...

Mary had a little lamb” heuristic – accentuate different words, change to synonyms or antonyms. Add “unless”, or ”...but without”
Change granularity – different levels of details give new perspectives
Sources for Test Ideas

1. Capabilities
2. Failure Modes
3. Quality Characteristics
4. Usage Scenarios
5. Creative Ideas
6. Models
7. Data
8. Surroundings
9. White-box
10. Public Collections

Each item promotes a way of thinking that can give you good test ideas. Some of these might not render any test ideas, but they can be very important to have in the back of your head.
The information sources are most useful when they are combined.
What I’m trying to tell you is that in order to do great testing, you must understand what is important, and you do that from a variety of information sources.

Never stop looking and learning.

This list will be released as a poster at
http://thetesteye.com/posters/TheTestEye_SourcesforTestIdeas.pdf
The potato is now filled with content from diverse sources.
Finale

• Drawbacks?
  – Difficult to stop
  – Requires trusting testers
  – Questioned by old-fashioned reviewers

• You have to find YOUR best ways

• Do your best, collaborate, learn to understand what is important

http://thetesteye.com/blog/

It is better to stop on purpose, than because you didn’t have any ideas.
If you don’t trust testers, train them, so you can.
I believe that in 10 years a medical audit might render questions like “So are you really saying you don’t do any exploratory testing to find out things you couldn’t predict??”

And there are no rules, and the few there are, you should sometimes break.
Questions

- ???

Further reading:
- Exploratory Testing Dynamics (Bach, Bach, Bolton)
- Styles of Exploration (Kaner, Johnson)
- BBST Exploratory Testing (Kaner, Bach)
- The Little Black Book on Test Design (Edgren)