These are my talking points for Usability Testing in Open Source Software, presented at Kieler Open Source und Linux Tage, September 2017.

On this slide, I’ll also talk about my background: open source software, and usability. I also teach usability for the U of M.

Tux mascot logo: Kieler Open Source und Linux Tage (fair use)
Original Tux mascot: Larry Ewing, 1996 (“Permission to use and/or modify this image is granted provided you acknowledge me lewing@isc.tamu.edu and The GIMP if someone asks.”)
CC BY

This presentation is distributed under the Creative Commons Attribution

Share — copy and redistribute the material in any medium or format

Adapt — remix, transform, and build upon the material for any purpose, even commercially.

Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

Also, my entire presentation is under the CC-BY.
What is usability?

Can real people do real tasks in a realistic amount of time?

Not the same as User eXperience (UX)

What do we mean when we talk about “usability”? You can find some formal definitions of usability that talk about the Learnability, Efficiency, Memorability, Errors, and Satisfaction. But I find it helps to have a “walking around” definition of usability.

A great way to summarize usability is to remember that real people are busy people, and they just need to get their stuff done. So a program will have good usability if real people can do real tasks in a realistic amount of time.

User eXperience (UX) is technically not the same as usability. Where usability is about real people doing real tasks in a reasonable amount of time, UX is more about the emotional connection or emotional response the user has when using the software.
You can test usability in different ways. You can directly measure (yes, you can measure usability!) in several different ways. I find the formal usability test and prototype test work well. You can also indirectly examine usability, such as using an expert to do a heuristic evaluation, or using questionnaires.

A note about the heuristic evaluation. This needs to be done by someone else, someone outside the project. It’s really hard for a developer to see the usability problems in their own programs (unless you do a direct test). In open source software, the developer creates the software. They spend a lot of time creating the interface. They know where all the menu items are and what they do. The labels and actions all make sense. But that may not be the same for someone who hasn't used the program before.
How to do a usability test

Who are the users? (personas)

Why are they using the system? (scenarios)

What are they doing on the system? (scenario tasks)

do until done {
    test;
    analyze results;
    tweak design;
}

It’s actually easy to do a usability test on your own. You don’t need a fancy usability lab or any professional experience. You just need to want to make your program easier for other people to use.

If you’re starting from scratch, you really have three steps to do a formal usability test:

1. Consider who are your users. Write this down as a short paragraph for each kind of user for your software. Make it a realistic fiction. These are your Personas. With personas, you can make design decisions that always benefit the user. “If we change ___ then that will make it easier for users like Jane.” “If we add ___ then that will help people like Steve.”

2. For each persona, write a brief statement about why that user might use the software to do their tasks. There are different ways that a user might use the software, but just jot down one way. This is a Use Scenario. With scenarios, you can better understand the circumstances when people use the software.

3. Now take a step back and think about the personas and scenarios. Write down some realistic tasks that real people would do with the software. Make each one stand on its own. These are scenario tasks, and they make up your actual usability test. Where you should write personas and scenarios in third-person (“___ does this..”) you should write scenario tasks in second-person (“you do this..”) Each scenario task should set up a brief context, then ask the tester to do something specific. For example:

   You don’t have your glasses with you, so it’s hard to see the text on the screen. Make the text bigger so you can read it more easily.

The challenge in scenario tasks is not to accidentally give hints for what the tester should do. Avoid using the same words and phrases from menus. Don’t be too exact about what the tester should do - instead, describe the goal, and let the tester find their own path. Remember that there may be more than one way to do something.

The key in doing a usability test is to make it iterative. Do a usability test, analyze your results, then make changes to the design based on what you learned in the test. Then do another test. But how many testers do you need?
You don’t need many testers to do a usability test - if you do it iteratively. You don’t get more useful results by adding more testers. Doing it with about five testers is enough to make tweaks to the interface. At five testers, you’ve uncovered about 80% of usability problems, assuming most testers can uncover about 31% of issues.

Note on this chart that zero testers uncovers zero usability problems!

But you may need more testers for other kinds of usability tests. “Only five” works well for traditional/formal usability tests. For a prototype test, you might need more testers.

And if you want to publish, you should really have something like ten or fifteen testers.

But five is enough for most things.

image: my own
based on Nielsen formula:
https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/
If every tester can uncover about 31% of usability problems, then note what happens when you have one, five, and ten testers in a usability test. You cover 31% with one tester. With more testers, you have overlap in some areas, but you cover more ground with each tester. At five testers, that’s pretty good coverage. At ten testers, you don’t have considerably better coverage, just more overlap.

image: my own
Who are the testers?

When you do a usability test, who should you ask to test the program? It depends on the program!

If your program is a program debugger, or an IDE, or some other programming tool, then you should ask other developers to be usability testers.

If your program is a solitaire game, then you should find people who play solitaire to test for you.

If your program is very general, and is intended for everyone, then you will have to include a variety of testers. Think about it as a two-dimensional problem. If it’s for everyone (like GNOME) then it should work well for all ages and all experience levels. So ages might be “10-20 years” and “20-30 years” and “over 30.” And experience might be “beginners” and “moderate” and “expert.”

image: my own
Let me show you a usability test that I did. Actually, I did two of them. This was part of my work on my Master’s degree. My capstone was *Usability Themes in Open Source Software*.

I wrote up the results for each test as a separate article for Linux Journal:

I like to show results in a “heat map.” A heat map is just a convenient way to show test results. Scenario tasks are in rows and each tester is a separate column.

For each cell (a tester doing a task) I use a color to show how easy or how difficult that task was for the tester. I use this scale:

—Green if the tester easily completed the task. For example, if the tester seemed to know exactly what to do, what menu item to activate or which icon to click, you would code the task in green for that tester.

—Yellow if the tester experienced some (but not too much) difficulty in the task.

—Orange if the tester had some trouble in the task. For example, if the tester had to poke around the menus for a while to find the right option, or had to hunt through toolbars and selection lists to locate the appropriate icon, you would code the task in orange for that tester.

—Red if the tester experienced severe difficulty in completing the task.

—Black if the tester was unable to figure out how to complete the task, and gave up.

There are some “hot” rows here, which show tasks that were difficult for testers: setting the font and colors in gedit, and setting a bookmark in Nautilus. Also searching for a file in Nautilus was a bit challenging, too. So my test recommended that the GNOME Design Team focus on these four to make them easier to do.
This is the heat map from my capstone project.

Note that I tried to do a lot here. You need to be realistic in your time. Try for about an hour (that’s what I did) but make sure your testers have enough time. Where you see the “o” in each cell is where we didn’t do that task.

You can see some “hot rows” here too: setting the font in gedit, and renaming a folder in Nautilus. And changing all instances of some words in gedit, and installing a program in Software, and maybe creating two notes in Notes.

image: my own
I’ve worked with others on usability testing since then. I participate regularly in Outreachy, formerly the Outreach Program for Women. Sanskriti, Gina, and Renata/Ciarrai/Diana. Allan and Jakub from the GNOME Design Team helped advise.

Outreachy logo: https://www.gnome.org/outreachy/ https://outreachy.gnome.org/ (fair use)
GNOME foot logo: https://www.gnome.org/logo-and-trademarks/ (fair use)
Most of the interns did a traditional usability test. So that’s what Sanskriti did here.

An internship cycle is thirteen weeks, structured like an online class.
Sanskriti did a usability test that was similar to mine, so we could measure changes. She had a slightly different color map here, using two tones for green. But you can see a few hot rows: changing the default colors in gedit, adding photos to an album in Photos, and setting a photo as a desktop wallpaper from Photos. Also some warm rows in creating notes in Notes, and creating a new album in Photos.

image: Sanskriti Dawle
Gina was from my second cycle in Outreachy, and she did another traditional usability test. This is a “formal” usability test.
Gina did a really good job that was very focused. You can see some hot rows here: bookmarking a location in Nautilus, adding a special character (checkmark) using Characters and Evince, and saving the location (bookmark) in Evince. Also some warm rows: changing years in Calendar, and saving changes in Evince. Maybe searching for a file in Nautilus.

Gina did such great work that we co-authored an article:

image: Gina Dobrescu (fair use)
In the next cycle, we had three interns: Gina, Ciarrai and Diana. Renata did a traditional usability test.
In Renata’s heat map, you can see some hot rows: creating an album in Photos, adding a new calendar in Calendar, and connecting to an online account in Calendar. And maybe deleting a photo in Photos and setting a photo as a wallpaper image in Photos. Some issues in searching for a date in Calendar, and creating an event in Calendar.


image: Renata Gegaj (fair use)
Ciarrai (pronounced same as “Kerry”) did a prototype test for a future change to the GNOME Settings application.
In the future Settings, the Design Team thought they’d have a list of categories down the side. Clicking on a category shows you the settings for that category. Here’s a mock-up for Wi-Fi in the new Settings. You can see the list of other categories down the left side.

image: GNOME, provided by Allan Day (fair use)
Remember the “only five” slide from a while back? That’s only for traditional/formal usability tests. For a prototype test, we didn’t think five was enough, so Ciarrai did ten testers.

For Ciarrai’s heat map, we changed the colors because the tester wasn’t actually using the software. They were pointing to a paper printout. Here, green indicates the tester knew exactly what to point to, and red indicates they pointed to the wrong one. Or for some tasks that had sub-panels, orange indicates they got to the first panel, and failed to get to the second setting.

You can see some hot rows, indicating where people didn’t know what category would have the Settings option they were looking for: Monitor colors, and Screen lock time. Also Time zone, Default email client, and maybe Bluetooth and Mute notifications.

image: Ciarrai Cunneen (fair use)
Debian

Traditional usability test

https://people.debian.org/~intrigeri/blog/posts/GNOME_and_Debian_usability_testing_201705/

And proof that others use the same usability test methods to examine usability: Debian did a usability test of GNOME. Here’s their test.
They had more general “goals” for testers, called “missions.” Similar to scenario tasks, the missions had a more broad goal that provided some flexibility for the tester. But not very different from scenario tasks.

You can see some hot rows here: temporary files and change default video program in Settings, and installing/removing packages in Package Management. Also some issues in creating a bookmark in Nautilus, and adding/removing other clocks in Settings.

image: intrigeri <intrigeri@debian.org> (Debian) (fair use)
More information

http://opensource-usability.blogspot.com/

Jim Hall
jhall@gnome.org

I hope this helps you to do usability testing on your own programs. Usability is not hard! Anyone can do it!

If you want more information, please visit my blog or email me.

On this slide, I'll also end for questions.
I don’t expect to use these slides. Only include if I have time.
For Diana, we wanted to do a UX test. This was the first time GNOME had done a UX test, and we were kind of figuring it out as we went. So Diana’s results aren’t very useful. But I figured I’d show them anyway.
Diana asked real people to use GNOME for the first time. They had three general scenarios to complete (access your email, use the web browser, “restore” your old files from this USB fob drive) to simulate how a new user would probably use GNOME in a real setting. That took about half an hour.

After that, Diana had them stop the test, and asked them what they thought. Think back to the first ten minutes, what did you think when you started using GNOME? Now that you’re done using GNOME, what do you think?

We had them point to an emoji that summarized their feeling, from “angry” to “love.”

image: Diana Kripak (fair use)
These are the results. It’s hard to show in a chart; this came out better as a prose description. Purple shows how the testers felt at the start of the test, and blue how they felt at the end. For most testers, the UX went up.

But we saw some differences in how people interpreted the emoji. I think if we were to do another UX test, we need to use fewer emoji that are more generally relatable, and we need to use more than five testers. (Remember, “only five” is when you are doing a traditional/formal usability test. You need more with a UX test.)

image: Diana Kripak (fair use)