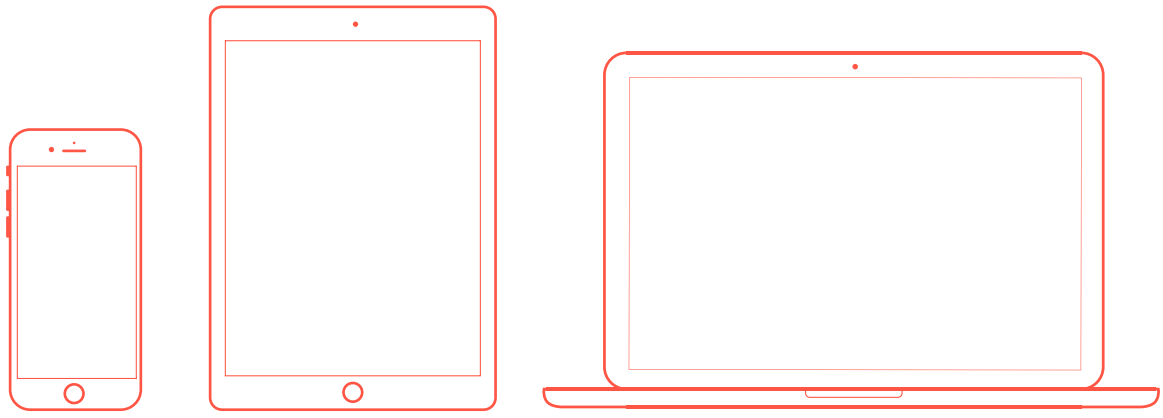
Workshop – Levi 10.4.2019

CompLeap project: pilot testing



This material was compiled to support the workshop at Levi on April 10th, 2019.

The first part at the beginning of each section is deliberately general and meant for familiarizing oneself with usability testing in its basic form, an evaluation method used in the context of software development. In these instances the objective is to find as many problems as possible in the application under study, and at best the development team starts fixing the problems even during the same day.

The latter part of each section is tailored for the CompLeap project. The purpose of this workshop is to build knowledge that helps the participants to pilot the prototype at poc.compleap.testiopintopolku.fi during Summer 2019.

In Helsinki, April 8th, 2019

Karri-Pekka Laakso
Anu Nikkanen

Usability testing

Usability testing is the most common and well-known method for finding usability problems from an application together with test participants. The test participants are given tasks which they try to complete independently. Most of the problems that arise with this method are learnability problems of first time use, i.e. problems understanding how the application should be used without instructions.

Usability testing is a reference method in “usability science”, as it is

- objective, because the problems arise from users’ actions and not from statements of experts, where the level of expertise influences the results,
- statistically significant, if there are enough participants, and
- repeatable and recordable (e.g. video, screencapture).

However, the former benefits have significance mainly in scientific work, as in real life

- the design expertise of the one who interprets the results influences the results a lot,
- statistically significant results require 20 or more users, but the most significant problems are uncovered with 4-6 users and observing the remaining test starts to feel like wasting your time, since the problems start to repeat, and
- recording videos is most cases a waste of time: it takes some effort to arrange, but watching the videos afterwards requires a huge amount of time and is hideously boring.

Sometimes usability tests are used as a checkpoint: where are we now, what needs to enhanced. However, the best impact is obtained when they are used in an iterative fashion while the application is under development: a short and lightweight set of tests uncovers most pressing problems, which are then fixed, and the test-fix cycle is repeated until the risk of remaining problems is reduced to bearable limits.

The key elements for a successful usability test are:

- the right test users,
- realistic test tasks, and
- a well facilitated test process.

Test users

Test users should resemble the real users of the application as closely as possible. Thus obtaining them often requires some effort: using only 30-year-old nerds from the development team does not comprehensively uncover problems from an application meant for everyone. On the other hand, to evaluate correctly an application meant for experts you need experts of the same type: doctors if it's meant for doctors, teachers if it's meant for teachers etc. - otherwise the results may turn out irrelevant.

Particular to CompLeap

Who are the pilot users?

How are the pilot users acquired? Is particular effort needed for finding the right kind of users?

What do we know about their background and their situation? What is needed for the pilot report?

Test tasks

Usability test tasks are realistical and typical ordinary situations written in the form of a test task:

1. You are buying yourself a tablet computer, because in two weeks you are travelling to Thailand and the flights take over 10 hours. Furthermore, it would be nicer to watch YouTube videos and browse the internet on a tablet than on your old Windows laptop. The trip to Thailand has reduced your budget down to approximately 600–700€. Buy yourself a suitable device.

The test task is thus a situation or a problem that the user should solve. It's essential that the problem is well balanced: not too hazy (the time runs out or it's hard to say when the task has been completed) nor too well-defined, like "Buy yourself the newest iPad" (unrealistically simple, contains pratically no decision making).

The best results are achieved when the test task is actually a real life use case obtained from the test user herself, like in the example above buying a tablet computer for herself. In those cases, the user is able to take into account all the realistic aspects that drive the decision making during the test. A predetermined test task can be in the worst case completely foreign to the user, e.g. the booking of a family trip given to a childless single test user, thus leading to unrealistic results.

Example test tasks

The following three test tasks are suitable for testing the train ticket service of Finnish Railroads (VR) at www.vr.fi :

1. You are travelling with your family (spouse, kids aged 2 and 5) to visit your distant relatives in Turku for the first time. You have agreed that you'd arrive at around 13:00, and you are returning back to Helsinki the same evening. Buy the tickets that are most suitable to your family.

2. You have promised to go and help your mom in Mäntyharju next Tuesday: She should visit an eye doctor at 10:30 (due to the eye-drops used in the examination, she can't drive home by herself), and she also needs some help in cleaning her apartment for the spring. Since you need to take one day off from work anyway, you are going to try to do at least some work during the trip. Buy the tickets for yourself for next Tuesday.

3. You and your spouse have tickets to Lauri Tähkä’s concert in Hämeenlinna at Verkatehdas on Friday 26.4.2019 at 19:00 The length of the concert is still a bit unclear, but it should end by 22:00. Since you live at a walking distance from the Tikkurila train station, you wonder whether you should go there by train? If it seems so, buy the tickets.

Particular to CompLeap

In what kind of environment and under what kinds of circumstances are the pilot tests carried out?

What is a good initial setting for these users?

This time, there is only one “test task”, which is to have the user to input his own educational background and her own selections into the application. The introduction to the tests should still be prepared beforehand. What should you say?

Facilitating the test

These are the most important tasks for the test facilitator:

1. Let the user do the tasks by himself: don't give advice, don't give hints, try to make the user feel like she is performing the tasks alone.
2. Make the user think aloud so that you understand what he is thinking and where do the problems arise from.
3. Make the users feel like you're not testing them. You are testing the application, and the possible problems are in the it and not in the test participant.
4. Somehow take notes of the actions which reveal the problems (see chapter Reporting).

First tell the user the following things:

- We are testing the system, not you.
- Please think aloud while doing the tasks. It helps us to understand what you are trying to do and what causes problems.

Explain the tasks to the user verbally and give them one at a time on paper, so that she doesn't need to remember them. The more the tasks resonate with the user's own reality the easier they are to remember, and short task descriptions are better than long ones. What you say aloud does not need to be exactly the same thing that you give to the user: you can give only the most essential facts in a list or **emphasize them** from the text.

The users will eventually forget to think aloud, especially once they start to wonder something. Remind them, even multiple times, if necessary: e.g. "Could you tell me what you are thinking right now?"

The users might ask for advice on what to do, but don't answer them directly. You can counter with a question, for example: "What do you think you have to do now? You can choose whichever option suits this case the best." If needed, you can even state "It's against the idea of this test to answer questions like these. You can just try to do what seems most reasonable for you."

As a facilitator, your job is not to

- get the user to "finish the task", or
- succeed in using the application,

but instead to

- get the user to do task all by himself,
- observe what happens,
- make sure that the user does not get a nervous breakdown. :)

Let the user do what they do, observe what's going on, remind them to think aloud and make notes.

Particular to CompLeap

If the piloting is carried out along other activities, like during a class or as a part of career counselling, it's counterproductive to emphasize that this is a **test**. A more natural approach is to say, for example: "Let's try out this application and see what thoughts it provokes."

In this context, the test facilitator does not need to say things like "We test the application, not you" since the idea of this being a test has not been told to the user at all. You two are just going to try out the application and see what happens. What is interesting is how the users start using it, what thoughts are provoked by the results shown by the application, and what he or she decides to do next (if anything). Just have the mindset of "trying out" instead of "testing", and everything will sort out by itself.

In this case, you do not need to give any test tasks on paper.

It is very important to emphasize thinking aloud.

Other things to remember:

Questions after the test

You can ask all kinds of things from the user after the test. You can even ask the user to fill in some kind of a questionnaire, though consider its value carefully especially when the amount of users is small. The most natural way is to ask the questions verbally and make notes of the user's answers.

The reason to ask questions is to gain an insight of the user's point of view of what happened in the test. The users tell certain kind of things when they think aloud during the test, but now they can share their interpretations of the events and how they understood the application and its concepts.

Though the users' answers provide depth and context, give most value to what they did and not on what they said. Users may tell you that something was fairly easy, though in fact they struggled with it for more than five minutes. The human mind is forgetful, and what they say is heavily influenced with the social context of the situation, things like an urge to succeed in a test or pleasing you by saying things they think you want to hear etc.

Particular to CompLeap

In this case, the questions asked after the test are exceptionally important, since the primary objective is to uncover higher-level issues than the usability problems of the prototype.

Questions after the test:

Reporting problems

An extrely lean way of working does not require practically any real reporting. Most of the problems are typically rather small issues with quite self-evident fixes, and thus they can be documented on post-it notes and discussed and prioritized immediately with the development team, for example. But even in that case it's fruitful to analyse what happened in the following format, though the intermediate results of the analysis would never be written down anywhere.

Write down the problems the users encounter and analyse them in the following format (the context in the example is an online gaming site):

> **Problem 1:** The user didn't notice that one can try out the games on the site without downloading and installing the client application.

Sum up the issue in one sentence. It should resemble a title. The shorter the better. This is often easiest to write after all the other parts.

> **Observations:** The user registered and immediately started installing the game client. After the installation the user complained that "It took quite a long time – these registrations and such.

You can't write this part unless you have some notes about the test. Thus, write down as many **observations** as you can: what did the user do? What did they write, where? What were they thinking, what did they click? Observations are more useful than your **interpretations** of what happened. Observations are facts, interpretations can be biased and they can change after receiving additional evidence. If you write down other things in addition to observations, mark them somehow to separate them later on from the observations.

You won't be able to make notes of everything during the test. That's ok. Making a video / screen capture including going through the recorded material is still far less efficient than working without them and missing some details.

> **Reason for the problem in the UI:** After registering, the site highlights the game client download with a big, colourful button, while the in-browser flash games are shown only with a small text link. From these options the user's attention goes to the client download button and the text link is easily missed. Thus the user might think that in order to play the games, he must install the client app.

This part is interpretation and analysis on the user behaviour, and it is done after the test. The problem and the suggestion of improvement can be

closely related, and sometimes it's even easier to comprehend the problem after writing the improvement.

Suggestion of improvement: <image of it>

Try to keep the improvement as local as possible. A total re-design shouldn't be necessary if the usability test is used at the right time for the right purpose: to find out mainly learnability issues after other methods have been used to weed out the problems of missing features, missing data, efficiency and cognitive load.

Particular to CompLeap

In what features of the prototype is the "observation-problem-improvement" analysis at its best?

What things you should not focus on in the case of this prototype?

What is the format for reporting the findings in the project?