Next Generation User Interface: Redefining file browsing

De Bleser, Jonas
jdeblese@vub.ac.be
Rollnumber: 0508848

Carraggi, Nicolas
ncarragg@vub.ac.be
Rollnumber: 0093262

Spruyt, Valentijn
vspruyt@vub.ac.be
Rollnumber: 0508466

December 23, 2015

Contents

1 Problem statement

The current state of file browsing is very similar on most desktop platforms. Based on an observation of the current file browsers, we conclude that each file browser consists of two important parts: a tree view and a content view. The first view shows the hierarchy and structure of the file system, whereas the second view simply shows the contents of one level in the hierarchy. We also note that browsing through these views is solely based on mouse and keyboard input. These observations exists for quite a long time and have not changed in the past decades. Therefore, our goal is to redefine file browsing by means of hand gestures, a flat file hierarchy and a more simple, intuitive interface. In the end, we are not really trying to solve a problem, but we rather want to create another approach towards file browsing.

Our approach consists of groups and tags to create a flat, nevertheless usable, hierarchy of files. A group is a general representation of a cluster of files. A tag is similar to a label and represents a keyword (e.g. vacation, Belgium, cat). A group is created by assigning it a name and several tags. Those tags determine which files will be related to this group. Assuming we add the tags `vacation` and `Belgium` to a group, it will cluster all files that have one of the aforementioned tags into that group. We decided to not support the feature of nested groups, because we want a clear overview of all groups. Nevertheless, some sort of nesting can also be achieved using our system. In terms of customizability, groups and tags can also have a color to characterize the appearance. Our idea was to utilize the gestures of the Myo to speed up the overall flow of managing a file system. Actions like cut, delete, etc would be mapped to a single gesture, reducing the amount of clicks from two (i.e. open context menu, click menu option) to one (i.e. simply do a gesture). Shortcut keys are, of course, another way to reduce the amount of clicks. Although, this still requires clicking the file, whereas with the Myo we simply hover to select. Unfortunately, this didn't go as expected as we explain in 4.2.

2 Requirements analysis

2.1 Usability requirements

2.2 Functional Requirements

The two main functionalities of our program consist of browsing and manipulating files, as well as providing a way of structuring folders in a custom way. The user should be able to browse the files easily using intuitive gestures. Next to browsing, we will introduce a new way of structuring files and folders, customized to the user its preferences. What follows next is a more specified way on how the file browsing will (probably) work.

Selecting files and folders will be based on the direction we point with our arm. Hovering a certain folder will result in a (possibly customizable) preview.

Browsing through the contents (i.e. files/folders) of a folder will be possible by rotating the arm and/or by doing the left and right swipe gesture to move backward and forward through the list of contents. To rename and search files we will use the keyboard as the primary input device, but we will also attempt to use voice recognition software as a supplementary input source to support people with a physical disability.

As far as customization of folders goes, we will attempt to make a more user-friendly way of grouping together files and folders, possibly using some sort of tagging system. In this way content specific folders are created and they will also have the ability to specify a custom view of the content. The concept of customization is still vague because we haven't decided yet on how we are going to implement it.

Manipulating the file system is done by performing an arm gesture, after which a menu similar to the one in figure 1. This menu will consist of the possible actions that can be executed on the selected file(s) and/or folder(s).

Navigation through the folder hierarchy is done by implementing some sort of breadcrumbs. Our initial thought was to use the up- and downward gesture in order to go up and down respectively in our tree of folders.

2.3 Data Requirements

The data we want to display and manipulate in our user interface comes straight from the underlying file system, i.e. information about files and folders. Our application should be able to access the file system without limitation. There has to be enough free storage space to store meta data of our application. This will be in the form of a database file which contains relations between groups, tags, etc.

2.4 Environmental Requirements

The environment must allow a person to easily move the arm of the user in front of the screen. The lighting does not matter since we don't use any recognition that is depend on external factors of the environment. We will also build a cross-platform application, therefor the user is not limited to a specific operating system to use the application.

1. **Description:** Checking the events per day should be possible within 2 steps starting from the main view

   Motivation: Since this is one of the principal aims of the system, it may not be difficult to do this. The user should be able to go to the next, previous or calendar day in one step, as well as applying a filter using another step.

   User class(es): User

Measuring concept: User satisfaction

Measuring method: Task scenario

- *Result:* Amount of steps to perform the task

Criteria for judging:

- *Worst level:* 3 or more steps
- *Planned level:* 2 steps - using the calendar as well as the filter
- *Best level:* 0 steps - The best possible level is zero because checking the events of the current day requires no additional steps

3 Design and interactions

Our approach to the interaction design is activity-centered design because we are focusing on the behavior of a file system.

3.1 Interactions

Instructing interactions is done mainly by the Myo. We define the following five gestures: `FIST`, `FINGERS_SPREAD`, `WAVE_IN`, `WAVE_OUT` and `DOUBLE_TAP`. We decided not to use the `WAVE_OUT` gesture because the Myo did not recognize this gesture very well. Also we noticed that, while performing the gesture, the position of the pointer drifted away quite far. The `DOUBLE_TAP` gesture is used to switch between the native mode and the group mode. Specifically for modals, we use `FIST` to accept and `WAVE_IN` to cancel. To open a menu on a specific item (i.e. file, folder, group and tag) we use the `FIST` gesture, which shows extra options that are applicable to that item. The `FINGERS_SPREAD` gesture on an item (and other interaction elements) performs the same operation as a normal click would do. Originally, we thought of using voice recognition to input text but it wasn't accurate enough to fulfill our requirements. Thus, the input interaction is done via keyboard.

3.2 Design

Since files, groups and tags are the most important part of our application, we decided to place them on the top of our application.

Nicolas Omer zei:

"interaction" was describing how the potential user would interact with the application. In our case, it is an Android application on a smartphone so it is via the touch screen, simple gesture, he uses tabs to navigate, push buttons, etc... (we explain all the things that have to be known to access every feature of the application)

So in our case that would be the 5 movements you have for the Myo I imagine + the fact that you can use another screen than the one of your computer, etc...

For the "design", we did pretty much the same than in the section "Style guidelines" for the project of the "User Interface Design" course (don't know if you have this one).

Anyway : we describe how the design process was conducted (how we came to such layout / display). Why a list view is better for storing your accounts (since you could have many), that we have drop-down menus to select a specific item in a list, etc...

Don't know if that is what they expect but it's what we have wink-emoticon

4 Technical report

Inleidend tekstje

4.1 Architecture

We based our architecture on Java to support multiple platforms. We have tested our application on both Windows and Mac to make sure our application works correctly. Metadata for our application is stored using HSQLDB[1] (HyperSQL DataBase). We also use Node.js[2] as a web server to serve files to the front-end. The user interface is build using HTML, CSS and JavaScript frameworks such as Bootstrap[3], Backbone.js[4], Marionette.js[5] and Require.js[6]. We made the decision to make the graphical user interface web-based, since it is a lot easier to customize the look and feel, as well as do advanced animations. To display the web content, we use JXBrowser which is a Java-based browser supporting Webkit[7]. Backbone.js is an MV* framework which provides us models, collections, views and additional functionalities to glue them together. However, it is very low level framework and therefore we use Marionette.js on top of it which gives us higher level components. For example, a `CollectionView` which provides us with all logic and functionalities to display a collection. In Backbone.js we would have to create such a common-used view from zero. We also use Require.js to work with dependencies for each file. In this way, only the required dependencies are fetched from the server, limiting the amount of requests to the server. However, once the application is finished, we can minify all dependencies to one file and simply load it using JxBrowser. As a result, we could remove the Node.js based web server entirely, but we left it for development purposes. Specifically, the only reason Node.js is used, is to avoid the same-origin policy problem with local dependencies. We also used a Java port[8] of the Myo SDK, instead of the default C/C++ SDK.

[1] http://hsqldb.org
[2] https://nodejs.org
[3] http://getbootstrap.com
[4] http://backbonejs.org
[5] http://marionettejs.com
[6] http://requirejs.org
[7] https://webkit.org
[8] https://github.com/NicholasAStuart/myo-java

4.2 Challenges

In this section we will discuss some of the challenges we encountered during the development of our project.

4.2.1 The Marionette.js framework

One of the biggest challenges of this project was the Marionette.js framework itself. Even though only one member of our team had experience with the framework, we all agreed on using it. The motivation for using this framework was that it has a neatly built model and view structure, which made it possible to write clean code that is fairly easy to read. The framework itself is a decent framework, but development went very slow because of the basic knowledge of JavaScript by the other two members. On the other side, by using this framework we increased our JavaScript skills, which will be handy for the future.

4.2.2 Myo

The Myo[9] bracelet is a something which can be used in a wide range of applications. When we watched a demo video, we immediately thought that it would be a nice thing to use for uit NGUI project. The nice thing about the bracelet is that it already supports several basic gestures and that it can detect spatial arm movement, next to just gestures of the lower-arm. However, when we started the project, we thought of adding custom gestures which were detectable by the bracelet. We saw fairly soon that this wouldn't be optimal since it was already difficult enough for the bracelet to detect the basic gestures. This was a big issue for the development of our project. When we tried to test our application with gesture `X`, the chances were that the bracelet detected gesture `Y` instead, which lead to great frustration. Another issue was the problem that comes with working with hardware, rather than software alone: Only one team member could take the Myo bracelet home and do some testing and programming with it. This caused delays in the development process, since it was very hard to synchronize work progress. An example: We decided to implement feature `A` in our application. One team member had set up the layout and views of the feature, another member coded some processing functions for the feature and the third team member was in charge of writing the code to send the myo gestures to our client side JavaScript. The problem was that, obviously, not everyone could work for the project 24/24 hours every day. This made development slower because we sometimes had to wait on another member to finish his assigned task. The other members couldn't just go and drive several tens of kilometers to go and fetch the Myo bracelet, so that they could write the code themselves. A last issue was that the Myo bracelet needs some warming up and synchronization before you can use it. This synchronization phase requires the user to perform a *wave-out* gesture for several seconds to even a minute long. This resulted into cramps in our lower arm more than once.

[9] https://www.myo.com/

TODO: NICO iets over calibratie ofzo?

4.2.3 JxBrowser

Tijd om license te krijgen responsiveness enzo, events doorsturen,

To bridge the gap between Java and JavaScript, we used the JxBrowser component. When we initially looked the component up, it looked very promising. A 30-day trial was available so we could start right away. After that we could get a free educational license, after filling in some forms. We soon realized that the JxBrowser component was rather processor-heavy, and that its responsiveness was sometimes rather low. The component came in the form of a *.jar* file, and modified something in the registry of our computer, so that we couldn't just use a new 30-day trial when the old one expired. This lead to several problems. In order to obtain an educational license, we first had to fill in a form with some information about the VUB. After that, we also had to ask Dr. Signer to send a recommendation mail to confirm that we are indeed students at the VUB. When the developers of the JxBrowser component finally sent us the license, it didn't work. There was some sort of bug that didn't change the registry key, so we kept getting the error that our 30-day trial had expired. By the time that we figured this out together with the developers, we had lost quite some time. Our initial motivation to use a web based user interface, was the ease to create animations. However, towards the end, we noticed that those animations were not that smooth as expected. We think that this is the result of the single threaded nature of JavaScript and the continuously sending of Myo events to the web browser.

5 Evaluation

5.1 Self-evaluation

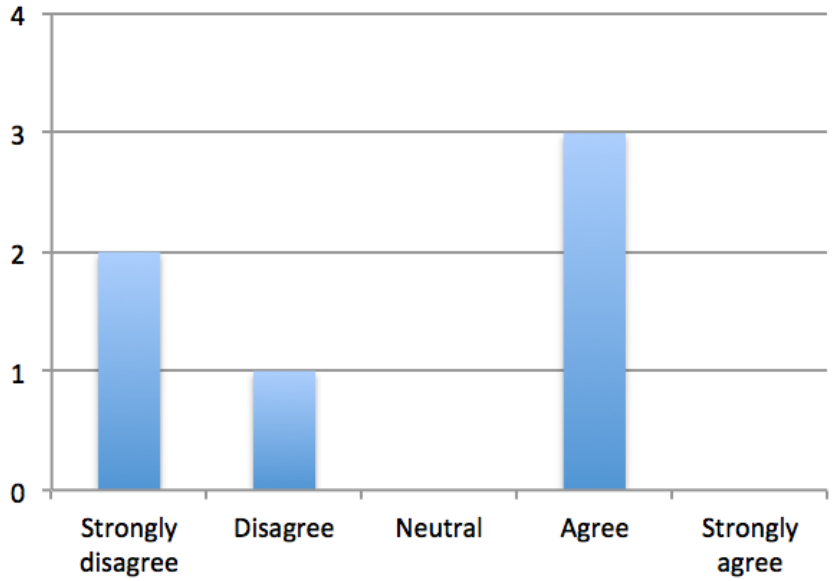
TODO ALOT OF TEXT

5.2 Evaluation by others

In this section we will discuss how others evaluated our project. First, we will take a look at the results from te questionnaire. After that, we will discuss these results by analyzing the feedback that we got from the evaluators.

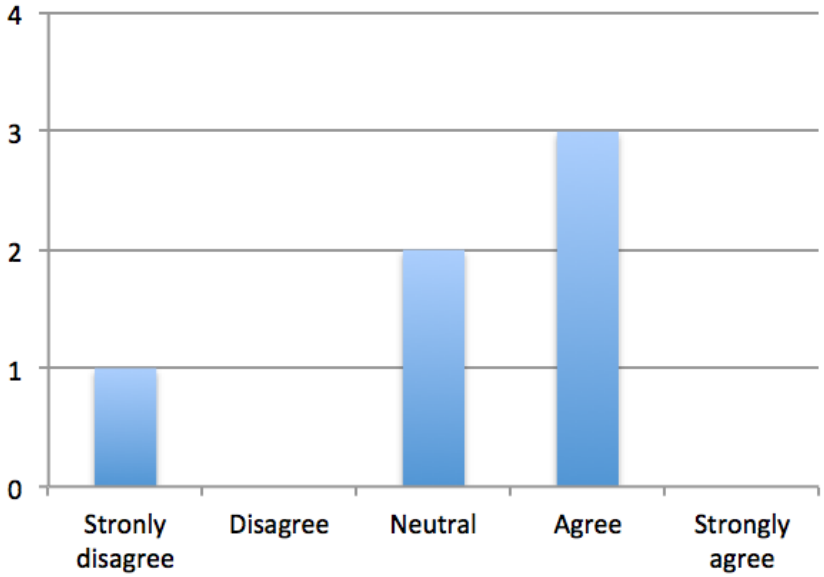
5.2.1 Questionnaire

Question 1: I think I would use the product often



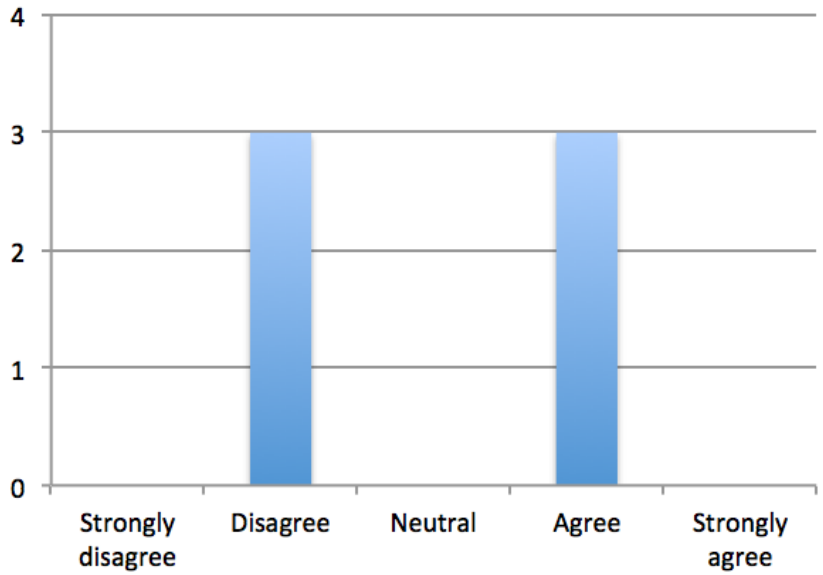
Graph 1: Answers to question 1

Question 2: I find the product unnecessarily complex



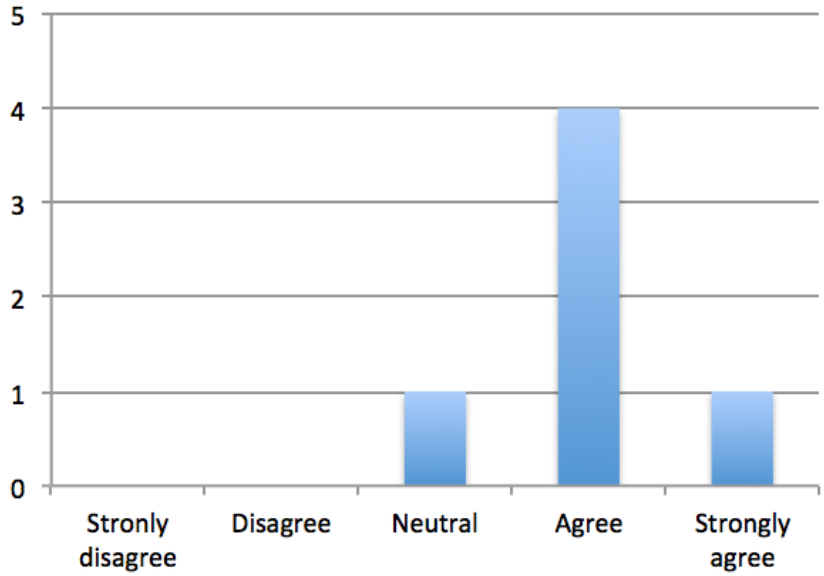
Graph 2: Answers to question 2

Question 3: I find the product easy to use



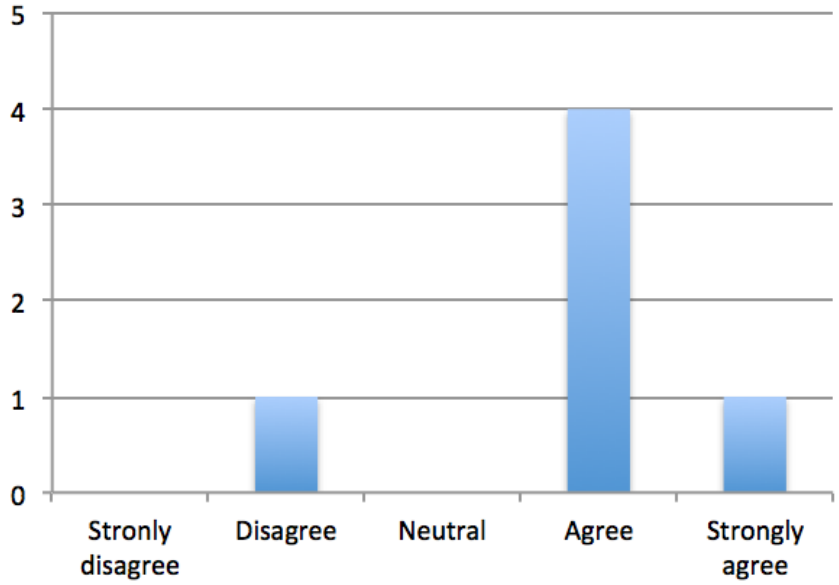
Graph 3: Answers to question 3

Question 4: I think I would need a technical person to use this product



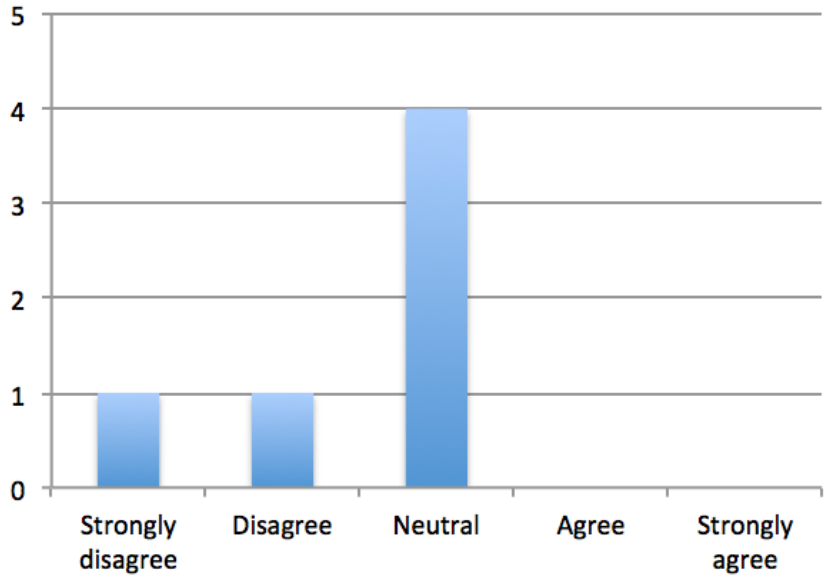
Graph 4: Answers to question 4

Question 5: I find that the different functions of the product are clear



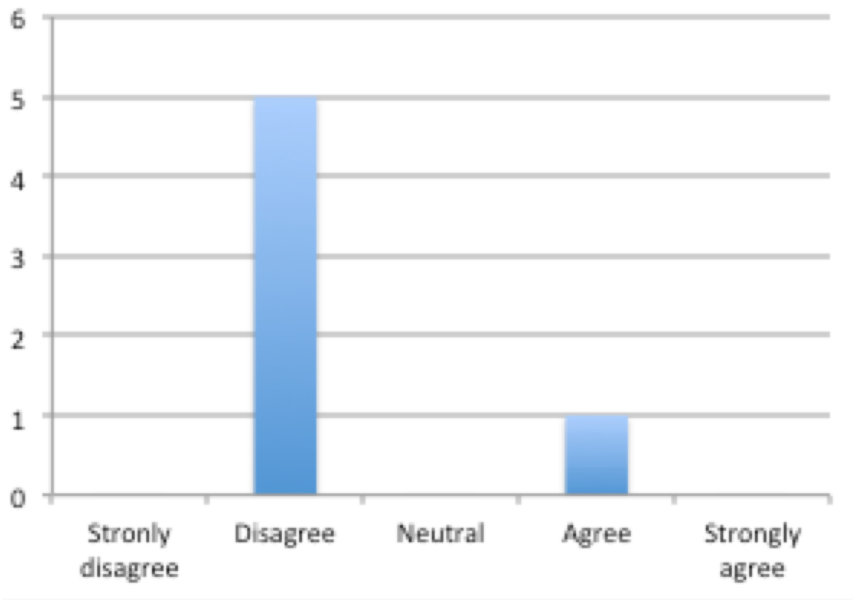
Graph 5: Answers to question 5

Question 6: I find the product inconsistent



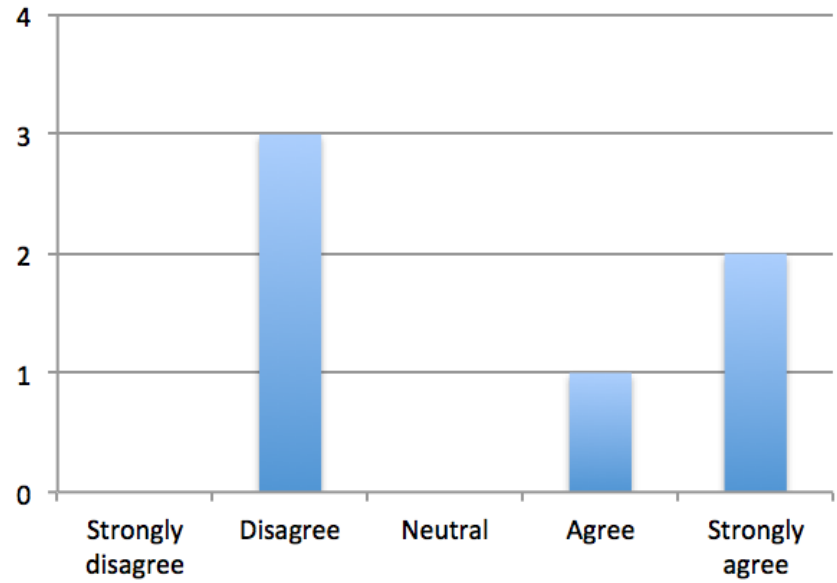
Graph 6: Answers to question 6

Question 7: I think most people will easily learn to work with the product



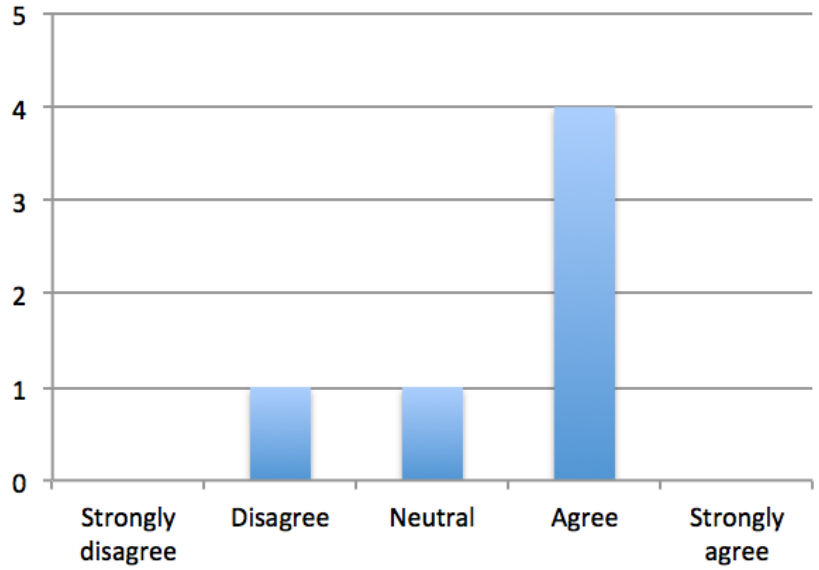
Graph 7: Answers to question 7

Question 8: I find the product awkward to work with



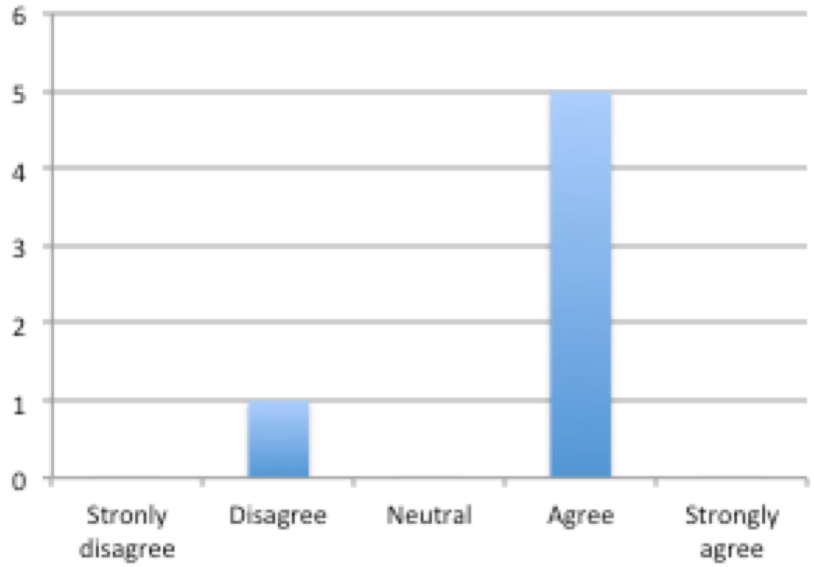
Graph 8: Answers to question 8

Question 9: I find it easy to see what functions are available when using the product



Graph 9: Answers to question 9

Question 10: I had to learn a lot before being able to use the product



Graph 10: Answers to question 10

5.2.2 Explanation

Once the design of the application was finished, we could start with a summative evaluation. We did this by using the observational evaluation technique talk aloud. We let six people, two parents and four friends, test our application by giving them several tasks and writing down how they used our application to achieve the goal of those tasks. An example task was to check the files that had a certain tag. We would then write down how the user navigated through the different screens. Another task was to try and create a group and then add atag to that group. Something noteworthy is that we asked the users to talk about what they are doing and if they were stuck, to tell us what they were searching for. The overall outcome of this evaluation was that our application is not ready for for real-world usage and that several improvements should be done before publishing the application. We will now discuss the answers to each question in more detail.

We see that the answers to the first question (I think I would use the product often) are very divergent. We think that this is because some people just really like the innovation that our project brings, but others might be used to the standard WIMP. When we asked for some feedback, we got two types of replies. The first type was that people didn't see the use of replacing the mouse with the Myo bracelet. This might be because using the Myo brings some overhead and it is also easy to get fatigue in your arm muscles by using it. The other type of reply was that they liked it because it was *something different*, but not necessarily better.

"I find the product unnecessarily complex". With this question we wanted to know if the combination of different gestures and the concepts of groups and tags wouldn't be too complicated for regular users. The answers made clear that most of the evaluators indeed found that our application was too complex. These evaluators are all used to a regular filebrowser and don't necessarily find the concepts of groups and tags beneficial to use. Only one of our friends was fully supportive of our project and saw the possibilities that groups and tags bring.

The opinions were divided for the third question. This question basically tested the usability of our application. Some people seem to struggle when testing our project, while others don't seem to have any problems with it. We do have to note that for some users, the Myo bracelet was detecting less gestures than for other users. This might have impacted the results.

Question four goes hand in hand with the second question. It seems that most people agree on needing a technical person to use the application. This somewhat baffled us because we didn't expect the results to be so one-sided. A possible explanation is that we needed to help each evaluator with synchronizing the Myo. This required the evaluator to perform a gesture for some time and afterwards we also needed to calibrate the Myo to detect the bounds of our application.

Next to the complexity of our application, we also wanted to know what the user thought of our application flow. Although the fact that the evaluators

found our application slightly too complex, they also found that the flow of the application was a good and logically structured flow. This wasn't that much of a surprise, since we are basically implementing the filesystem all over again, but with some conceptual and design changes.

When asking if people found our application inconsistent, most of the evaluators didn't really have an opinion. This might be because they didn't fully understand the question. What we wanted to know, was that when a user wants to perform some task, he has to perform a sequence of actions. If he then wants to perform a similar task, he should also perform a similar action sequence. The evaluators who did understand the question, all answered that our application is consistent.

The overall outcome of question seven is that our application would be hard to learn. Although we don't agree on this, we do understand why people might think in this way. When we asked these people to evaluate our application, they all had to learn the gestures and functionalities in a limited timeframe. This probably influenced them in a negative manner, resulting in these answers.

"I find the product awkward to work with". This question was based on the fact that people do have to make strange gestures in front of their screen, which might be a bit awkward. With awkward, me meant that the user might feel uncomfortable when using the application, but also that the user did not find it handy to use the Myo bracelet. 50% of the evaluators were uncomfortable when using the application. This is a rather high percentage, which might indicate that the world isn't ready to combine working on a computer and using gestures to control that computer.

We attempted to display as much relevant information as possible on the screen, which resulted in a positive evaluation for question nine. Users seem to know what the enabled tasks are when they use our application. We are very happy that this is the case, since using a filebrowser should be intuitive and should not take that much time to perform a certain task.

Question ten is very similar to question seven. We again suspect that the limited timeframe to learn and understand the different gestures and functionalities might have influenced the different evaluators. We could argue that these answers might be a bit too subjective. When they first introduced a mouse as a pointer, people also had to learn how to use it and what the different mousebuttons did.

Source code (December 23): The source code of your next generation user interface together with detailed and complete instructions on how to run it (i.e. step by step instructions together with possible dependencies, required software and technologies, etc.). Note that your source code should be documented: every important method and class should be annotated with a description of its goal and functionality. If you use a tutorial, open source framework or other online code, you have to mention this and cite the author's website. We check for code plagiarism! The source code has to be sent to Sandra Trullemans (strullem@vub.ac.be).