1. Theoretical framework
   1. Usability in the software process

Performing a user-centered system design as a part of the overall software process means that you will need to implement a user-centered requirement framework in the selected software process. Zimmerman and Grötzbach suggest one framework where they introduce three types of non-functional requirements, usability requirements, work flow requirements and user interface requirements.

Usability requirements are..

Work flow requirements are a description of how the software system is supposed to support the user when trying to perform certain tasks. They are a description of what actions the user will need to take and how the system will achieve these. The actions needed to be performed can be described in analysis artifacts such as use cases or scenarios. These requirements can when the system is finished be used to evaluate the usability of the system and the requirement itself can be validated and see that the actual work flow corresponds to the requirement.

* 1. User interface requirements define how the interface should look like and translate into design artifacts such as sketches, navigation models, information architecture and eventually paper prototypes.[[1]](#footnote-1)
  2. Why agile user-centered methods?

Agile methods are the name for the group of software development processes where the project is divided into shorter iterations or sprints. Each iteration may last a few weeks where in the beginning goals for that iteration are defined and in the end the result is presented for the managers. During the sprints most of the phases of the software process are gone through, requirements are analyzed, designed for and implemented. This way of working has several advantages, first of all, in a rapidly changing environment that the software system is created new requirements may appear and old ones might be in the need of change. Following a plan based approach; months of work might be lost because of the huge amount of initial planning.

The key is in other words rapid and dynamic development which is also in line with what suitable for processes where the end users are involved, user-centered development processes.

* 1. Phase
  2. Release planning phase and sprints

In the release planning phase the sprint planning is done, first of user stories are created on so called story cards. These describes requirements for the systems from the user's perspective. The description consists of what the user want to do and for the development team this description is all that matters and any other restrictions or demands on the task needs to be specified in their own user story. In other words stories will be short and this serves another purpose in that they can be implemented quickly which is a great advantage in the agile process. Developers can implement a new story within hours or days and if there are problems which renders the story obsolete not much time has been wasted.

After all story cards are created the sprint planning can start, usually a sprint will go on for a few weeks and this includes both implementing and testing. Each sprint will be allocated a certain amount of stories where the number depends of the complexity of the stories. The planning should makes sense in that any stories that depends on other stories need to be scheduled after their dependencies. Furthermore, stories that are more important should be implemented early on and so should stories that are dependent on complex technology. In the case of issues with that particular technology there is then still time to fix these compared to if those stories would have been pushed back to end of the development phase.[[2]](#footnote-2)

* 1. Methods for evaulating usability

Evaluation techniques are often grouped into two categories, namely expert evaluation and participant-based evaluation. The latter means that end users or a group representing them will be a part of the evaluation, mostly by actually using the system and having to answer questions or surveys. Expert evaluation means that the system under design is evaluated by usability experts. This method however should never be used by the designers themselves as they could have significant bias towards the system as they know it very well and could potentially both find too few problems or obscure problems that aren’t realistically going to occur during regular use. [[3]](#footnote-3)

One example of participant-based evaluation is cooperative evaluation. This means that a user will try performing predefined tasks (these tasks should of course be part of realistic future use) together with the expert performing the evaluation. All this could be video- or audio recorded to get the most out if it but it can also be sufficient that the expert is taking notes on how the program is performing. During the process the participant will be encouraged to talk out loud and the expert will also be asking a series of questions. For a detailed description of guidelines for a cooperative evaluation see table 1.[[4]](#footnote-4)

1. Table - Guidelines for cooperative evaluation. [[5]](#footnote-5)

|  |  |
| --- | --- |
| Step | Notes |
| Using scenarios prepared earlier, write a draft list of tasks. | Tasks must be realistic, do-able with the software and explore the system thoroughly. |
| Try out the tasks and estimate how long they will take a participant to complete | Allow 50 percent longer than the total task time for each test session |
| Prepare a task sheet for the participants | Be specific and explain the tasks so that anyone can understand |
| Get ready for the test session. | Have the prototype ready in a suitable environment with a list of prompt questions, notebook and pens ready. A video or audio recorder would be very useful here. |
| Tell the participants that it is the system that is under test, not them; explain and introduce the tasks | Participants should work individually – you will not be able to monitor more than one participant at once. Start recording if equipment is available. |
| Participants start the tasks. Have them give you running commentary on what they are doing, why they are doing it and difficulties or uncertainties they encounter. | Take notes of where participants find problems or do something unexpected, and their comments. Do this even if you are recording the session. You may need to help if participants are stuck or have them move to the next task. |
| Encourage participants to keep talking. | Some useful prompt questions are provided below. |
| When the participants have finished, interview them briefly about the usability of the prototype and the session itself. | Some useful questions are provided below. If you have a large number of participants, a simple questionnaire may be helpful. |
| Write up your notes as soon as possible and incorporate into a usability report. |  |
|  |  |

7 Evaluation

* 1. Choice of evaluation method

FlyTracker is mainly aimed at a few users at the motion vision lab at Uppsala University

* 1. Process

The

* 1. Result

**2.3.2 Sequence models**

*A sequence model* is a model describing user strategies for performing tasks, the model defines the intent and the steps required to perform a task. The model describes in detail exactly what the user currently does when solving certain problems. Modeling this gives an overview of what is actually happening and what potential issues there are. A sequence model is therefore a good tool to possibly find better strategies than the ones currently employed by the users.[[6]](#footnote-6)

**2.3.3 Scenarios**

There are different types of usages for scenarios, such as conceptual scenarios. A conceptual scenario is an abstract description of how a user performs a specific task There is no detailed description of tools and technologies used. nor suggestions for how the tasks should be solved. Using conceptual scenarios is a good way to generate requirements as it leads to concrete scenarios which are descriptions with high level of detail. These will provide suggestions for the future interface design (Benyon 2010).

1. Zimmerman, D, Grötzbach, L *”A Requirement Engineering Approach to User Centered Design”* [↑](#footnote-ref-1)
2. Beyer [↑](#footnote-ref-2)
3. Benyon, s.228-232 [↑](#footnote-ref-3)
4. Benyon, s.232 [↑](#footnote-ref-4)
5. Benyon, s.232-233 [↑](#footnote-ref-5)
6. Beyer, Hugh, *User-Centered Agile Methods* [↑](#footnote-ref-6)