**“User-Centered Process for Designing and Implementing Interactive Motion Tracking Software”**

**Introduction**

**Purpose**

**Methodology**

**Tidigare forskning**

Teorin behandlar främst frågor som vad det är som kännetecknar en ACSD och varför en sådan har valt. Dessutom diskuterar den de verktyg som finns för att utvärdera ett system och vilka mått det finns på användarvänlighet.

Ett sådant mått som kan användas i en utvärdering som är passande för ett projekt av denna typ är som jag redan nämnt *”usability”*. Vad som avses med det är helt enkelt att programmet ska kräva så lite arbete som möjligt av användaren för att utföra de uppgifter som systemet är tänkt att kunna göra. Det ska förse användaren med korrekta funktioner med nödvändig information och att de är organiserade på ett naturligt sätt i det grafiska gränssnittet. Att mjukvaran är lätt att lära sig och att saker och ting finns där de förväntas finnas är också ett kännetecken för *”usability”.[[1]](#footnote-1)*

För det sistnämnda är *”design principles”* viktiga då det kan underlätta inlärningsfasen genom att applicera vanliga *”design principles”.* Vad som avses då är artefakter som ofta hittas i liknande system, t.ex. ångra/gör om-knappar, att alltid visa för användaren vad som pågår, placera knappar och information på ställen där de hör hemma. [[2]](#footnote-2)

Fokus på ”usability” tenderar ofta att försvinna i delar av mjukvaruprocessen av främst två olika anledningar. För det första handlar det ofta om relationen till kunden där kunden sällan specifikt poängterar vikten av det och när företaget sen levererar produkten till kunden får den en mindre användarvänlig produkt. Här behövs frågan diskuteras tydligare då kunden kanske inte ens känner till konceptet ”usability” även om det på abstrakt nivå är något den önskar. Ett annat problem som kan uppstå är när ”usability”-experter bara är delaktiga i delar av systemutvecklingsprocessen så kommer ”usability” bli lidande i de faser där de inte är delaktiga (vanligtvis implementerings och testningsfasen). Anledningen att de inte är delaktiga är att MDI-experter sällan också är programmerare och därmed är det rena datavetare och systemutvecklare som oftast genomför denna del av utvecklingsfasen. Det som behövs är HCI-experter som har tillräckligt god kunskap om systemutveckling för att kunna delta i implementeringsfasen och ge sin input där. [[3]](#footnote-3)

Det finns två grupper av utvärderingsmetoder, dels har vi expert-utvärderingar där MDI-experter använder olika tekniker för att utvärdera ett givet system eller design. Den andra är användarinvolverade utvärderingsmetoder. Dessa låter antingen de faktiska slutanvändarna, eller om det inte är möjligt någon som kan representera dessa, testa prototypen eller det färdiga programmet. Det finns sedan många metoder för att få ut ett resultat från denna utvärderingsmetod, man kan observera användarna under testets gång eller låta de fylla i formulär eller svara på frågor via intervju. Både expertutvärderingar och användarinvolverade utvärderingar har för- och nackdelar men har man tillgång till slutanvändarna finns de få anledningar att inte låta de testa systemet. Expertutvärderingar ska heller inte göras av den egna designen då den djupgående kunskapen om systemet gör att bedömningen riskerar bli snedvriden. Antingen kan problem som en utomstående expert hade hittat inte dyka upp i utvärderingen eller så hittas obskyra problem som realistiskt i princip aldrig kommer dyka upp i faktiskt användande.[[4]](#footnote-4)

En typ av användarinvolverad utvärdering är kontrollerade experiment vilket är en mer strikt form av prototyptestning där vad som ska utvärderas är tydligt definierat och kvantifierat. Det kan vara till exempel vara jämförelser av två olika gränssnitt där man vill hitta det mest användarvänliga.

I metoddelen beskrevs redan delar av vad en agil ACSD innebär. Mer i detalj så består hela processen av en fas 0 där grupper av användare identifieras och studeras. Vad är det slutanvändarna kommer att göra med programmet och vad vill de få ut? Detta görs genom ett antal metoder, bland annat definierar man så kallade ”personas” vilket då är en generell beskrivning av en subgrupp av slutanvändarna. Denna beskrivning kan sen användas för att designa för specifika önskemål eller problem som skulle kunna uppstå. Man tar fram ”user stories” och scenarion för att visa på typiska användningssituationer. Allt detta görs genom olika typer av domänmodeller, sekvensmodeller beskriver till exempel olika arbetssekvenser för olika funktionaliteter i programmet. Hur hanteras det och vad är det som krävs av användaren för att genomföra en given uppgift.[[5]](#footnote-5)

Pappersprototyper är också ett bra sätt att tidigt i utvecklingsfasen designa ett optimalt grafiskt gränssnitt genom att enligt användarnas önskemål skapa en prototyp för hur gränssnittet ska se ut och sen låta användarna prova på det. Det går snabbt att modifiera en sådan och användarna kan testa sig fram till vad de föredrar.[[6]](#footnote-6)

**Agile methods**

Agile methods are the name for the group of software development methods where the project is divided into shorter iterations. 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.

**User-centered design in the agile process**

**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 artefacts 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 workflow corresponds to the requirement.

User interface requirements define how the interface should look like and translate into design artefacts such as sketches, navigation models, information architecture and eventually paper prototypes. [[7]](#footnote-7)

**Methods for evaluating usability in software**

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. [[8]](#footnote-8)

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.[[9]](#footnote-9)

Table 1 - Guidelines for cooperative evaluation. [[10]](#footnote-10)

|  |  |
| --- | --- |
| **Step** | **Notes** |
| 1. Using scenarios prepared earlier, write a draft list of tasks. | Tasks must be realistic, do-able with the software and explore the system thoroughly. |
| 1. 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 |
| 1. Prepare a task sheet for the participants | Be specific and explain the tasks so that anyone can understand |
| 1. 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. |
| 1. 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. |
| 1. 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. |
| 1. Encourage participants to keep talking. | Some useful prompt questions are provided below. |
| 1. 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. |
| 1. Write up your notes as soon as possible and incorporate into a usability report. |  |

1. Benyon, David, *”Designing Interactive Systems”,* s. 84 [↑](#footnote-ref-1)
2. Ibid, s.89 [↑](#footnote-ref-2)
3. Göransson, B, Gulliksen, J, Boivie, I, *”The Usability Design Process – Integrating User-Centered Systems Design in the Software Development Process”* [↑](#footnote-ref-3)
4. Benyon, D, *”Designing Interactive Systems”, s. 228-235* [↑](#footnote-ref-4)
5. Beyer, Hugh, *User-Centered Agile Methods* [↑](#footnote-ref-5)
6. ibid [↑](#footnote-ref-6)
7. Zimmerman, D, Grötzbach, L *”A Requirement Engineering Approach to User Centered Design”* [↑](#footnote-ref-7)
8. Benyon, s.228-232 [↑](#footnote-ref-8)
9. Benyon, s.232 [↑](#footnote-ref-9)
10. Benyon, s.232-233 [↑](#footnote-ref-10)