SE Software Engineering
The University of Auckland

Software Development Methodologies

Lecture 8 - User Studies 1

SOFTENG 750 2013-04-29

User Studies Overview

If we knew what it was we were doing, it would not be called research, would it?
(Albert Einstein)

User Studies

How do we know your software is useful?

- Software Engineers are biased: we like our own software
- Your friends are also biased

Solution: conduct a user study

1. Define **research questions** and **hypotheses**
2. Choose a **methodology** to investigate them
3. Recruit participants and collect **data** from them
4. Interpret your data to find answers and verify if hypotheses are true

Without a user study you cannot be sure whether your software is great or not.

Qualitative vs Quantitative User Studies

Qualitative Studies

- Show users a (paper-) prototype and get feedback
- Possibly let them use the prototype (e.g. specific tasks)
- Interviews, open questions, think-aloud protocol, observations, ...
- Data typically in the form of text: statements of users, observations, ...
- Good to explore an early prototype:
  What are the problems with this UI?

Quantitative Studies

- Tasks or questionnaires with measurable outcomes (data are numbers)
- Fairly rigorous methodology:
  variables, hypotheses, measurements, statistics...
- Good to compare UIs / apps:
  Which UI is better? How much better?

Often mixed-methods approach: quantitative & qualitative

How to Conduct an Empirical Study

1. Choose **research questions**
   - ○ Specific enough to be answerable, general enough to be interesting
   - ○ Specify the target population

2. Create a study **design**
   - ○ Define tasks (i.e. what do users do during the study)
   - ○ Define independent and dependent variables and specify how they are measured
   - ○ Define hypotheses based on the variables
   - ○ Create a **script** for the study (step-by-step guide)

3. Conduct a **pilot study** and revise design & script

4. Recruit participants, use script to **collect data**

5. Analyze the data, test hypotheses, interpret & discuss

Variables

Independent Variable (IV): What do I change?

- Variable of which we want to know the effect
  - We change it (try different values and see what happens)
- **Levels**: the different values that we try out & compare
  - E.g. the UI used (list view vs gallery view, different menus...), other parameters (big vs small screen)
  - Usually only few independent variables and levels
  - Levels lead to **conditions** that need to be tested
  - More conditions means more time required

Dependent Variable (DV): What do I observe?

- Variable that describes the **effect** we are investigating
- Needs to be **measurable** (as accurately as possible)
  - Can be many if measuring them is cheap (more data is good)
  - E.g. completion time, questionnaire score

Types of Empirical Studies

Controlled Study:

- **Change** the independent variables (try different values)
- **Measure** the dependent variables (to find out about effects)
- **Keep** everything else the same as much as possible

Observational Study:

- Variables are not controlled, but merely observed
- Try to infer effects from the observed values
- Sometimes necessary because variables can be difficult to control (e.g. weather, user behavior in big organization)

Lab vs Field Study:

- **Lab**: More control (good for controlled studies), less "contamination" by uncontrolled variables
- **Field**: Less control (i.e. usually observational), more realism

Requirements

Ethics Approval: is the study ethical?

- Most big organizations require an approval process
- Possible problems: damage, power abuse, deception (often used, but needs to be handled responsibly)
- Standard practice: give participants **info sheet** to read, then ask them to sign a **consent form**

Participants: how to recruit people from the target population?

- Advertise in the right places (often low response rate)
- Motivation?
  share results, reward (money, voucher, food etc.)

Environment & Equipment:

- Quiet space, controlled and consistent lighting (e.g. lab)
- Computer, software, eye tracker, camera...

User Study Design

Defining Tasks

Tasks of a controlled user study should be...

- **Relevant** for answering the research question
- **Well-defined**: it should be clear for a participant what to do
  - Have a clear starting point, clear goal
  - Avoid choices, otherwise the data gets erratic
  - Make it easy to train unfamiliar participants
- **Realistic**: how do real users do it?
  - Make results generalizable to the real world
  - Find balance with well-definedness/simplicity
- **Variations**: find similar tasks that test exactly the same thing
  - Need tasks for training, different levels of IV, repetitions to generalize results to a whole use case (not just the task)
- **Well-timed**: how long does the task take?
  - Prevent data gathering from getting too time consuming
  - Ensure task doesn't get out of hand (consider worst case)

For qualitative & observational studies task definitions less important

Measuring Usability

Performance

- *Task completion time*, operation counts, *eye gaze path length*
- Specific performance scores*:* e.g. productivity scores

Accuracy

- Number of mistakes: define what exactly counts as a mistake
- Or measure deviation from an objective optimum
  (e.g. "align objects perfectly", "find cheapest price")

Satisfaction

- Subjective but often more important than performance
- Typically measured with questionnaire
  (e.g. Likert-scale, "I enjoyed using the system")

Measuring DVs

Define precisely how you measure your dependent variables.

Quantitative Measures

- *Task completion time*: define start and finish events
- *Event counts:* key strokes, mouse clicks, mouse path, keyboard/mouse switches (usually from event log)
- *Eye gaze path length* (from eye tracker)
- 5-point Likert-scale items with standard labels (subjective!) "strongly disagree" to "strongly agree"
  - Common Likert-scale labels: http://www.gifted.uconn.edu/siegle/research/Instrument%20Reliability%20and%20Validity/Likert.html
  - http://dataguru.org/ref/survey/responseoptions.asp

Qualitative Measures

- Open questions, e.g. "What did you dislike about the system?"
- Think-aloud protocol statements
- Observations, e.g. observed participant comments/reactions
- Eye gaze pattern

System Usability Scale

Measures subjective usability with standard 5-point Likert scale:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

See also http://www.measuringusability.com/sus.php

Measuring Demographics

Make sure to collect **demographic data**

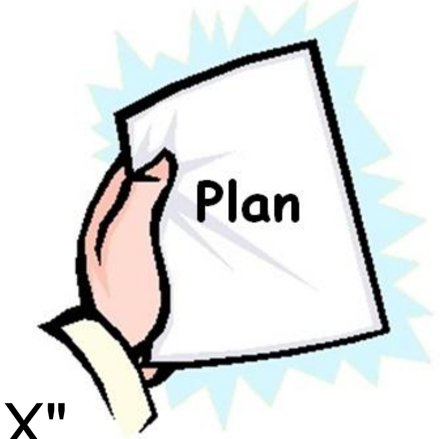
- Data describing your sample of users (e.g. gender, age, occupation)
- They may have an effect on the DV
- Effects on the DV can be analyzed later (very interesting, e.g. gender differences)

Important to discuss whether the results can be generalized to the whole population of interest (not just SoftEng students)

Use questionnaire:

1. Age, Gender, Occupation
2. Relevant experience (specific to your tasks), e.g.
   - Have you used a similar application before? How much?
   - How many hours weekly do you spend playing computer games?
   - "I do a lot of word processing in my everyday computer use" (Likert-scale)

Example Study Design 1



1. **Research Questions:**
   "What are usability problems of our app?"
   "Which features are the most important?"

2. **Tasks**: what do the participants do? "accomplish goal X"
   - At least one task for each of the features

3. **Measurement**
   - Observe mistakes made during tasks
   - Record think-aloud protocol
   - Questionnaire at the end
     (*demographics, satisfaction, ranking of features*)

4. **Schedule**: who does what, when, how often?
   - Possibly let participants decide what tasks they do
   - Possibly change task order between participants

Example Study Design 2

1. **Independent Variable**
   Two apps: your app (*A*) and competitor (*B*)
2. **Dependent Variables**
   performance, *accuracy*, *satisfaction*

3. **Hypotheses**: what outcome do we expect
   "*A* has better performance, accuracy & satisfaction"
4. **Tasks**: what do the participants do? "accomplish goal X"

5. **Measurement**: how do we measure the dependent variables?
   Task completion time, count mistakes, questionnaire at the end (*demographics* & *satisfaction*)
6. **Schedule**: who does what, when, how often?
   Group1: 3 tasks with *A*, then 3 tasks with *B*
   Group2: 3 tasks with *B*, then 3 tasks with *A*

Today’s Summary

- **Controlled studies** investigate the effect of independent variables on dependent variables
- Studies have a **design** and are defined in a **script**
  - ○ **Tasks** need to be specified (using certain criteria)
  - ○ **Measurement** methods need to be specified

Interim Report due Today 7pm

In a subfolder "reports" in the root folder of your repository.

Milestone (Deadline: Lab on Thursday)

1. Design a usability study for your app
2. Create a script for the study

Quiz

1. What are independent and dependent variables?
2. Describe the criteria for defining good user study tasks.
3. What is the system usability scale?

```
               /*                                  ,*/
              #include                           <time.h>
               #include/*                  _  ,o*/  <stdlib.h>
               #define c(C)/*    -      . */return    ( C); /*   2004*/
                #include  <stdio.h>/*.  Moekan         "'   `\b-'    */
                  typedef/* */char  p;p* u                   ,w      [9
                    ][128] ,*v;typedef int _;_ R,i,N,I,A             ,m,o,e
               [9],  a[256],k   [9], n[             256];FILE*f     ;_ x  (_ K,_ r
           ,_ q){;  for(;                                    r<    q    ; K      =((
        0xffffff)  &(K>>8))^                                   n[255    &       ( K
      ^u[0       +                                  r ++      ]    )]);c         (K
     )}      _ E                        (p*r,   p*q ){    c(         f        =
            fopen                  (r  ,q))}_  B(_ q){c(   fseek      (f,     0
           ,q))}_ D(){c(  fclose(f ))}_  C(  p    *q){c(  0-    puts(q     )   )}_/*   /
         */main(_ t,p**z){if(t<4)c(  C("<in"      "file>"   "\40<1"  "a"  "yout> "
        /*b9213272*/"<outfile>"  ) )u=0;i=I=(E(z[1],"rb")) ?B(2)?0 :  (((o  =ftell
       (f))>=8)?(u   =(p*)malloc(o))?B(0)?0:!fread(u,o,1,f):0:0)?0:  D():0    ;if(
      !u)c(C("     bad\40input  "));if(E(z[2],"rb" )){for(N=-1;256> i;n[i++] =-1   )a[
      i]=0;      for(i=I=0;  i<o&&(R  =fgetc(  f))>-1;i++)++a[R] ?(R==N)?( ++I>7)?(n[
     N]+1      )?0:(n [N  ]=i-7):0:  (N=R)    |(I=1):0;A =-1;N=o+1;for(i=33;i<127;i++
     )(      n[i  ]+  1&&N>a[i])?   N= a    [A=i]     :0;B(i=I=0);if(A+1)for(N=n[A];
    I<      8&&  (R  =fgetc(f ))>   -1&& i  <o        ;i++)(i<N||i>N+7)?(R==A)?((*w[I
    ]            =u [i])?1:(*w[I]=  46))?(a           [I++]=i):0:0:0;D();}if(I<1)c(C(
                " bad\40la" "yout  "))for(i          =0;256>(R=  i);n[i++]=R)for(A=8;
               A >0;A --)  R  = ( (R&1)==0)         ?(unsigned int)R>>(01):((unsigned
              /*kero Q'        ,KSS  */)R>>       1)^       0xedb88320;m=a[I-1];a[I
              ]=(m          <N)?(m=  N+8):      ++        m;for(i=00;i<I;e[i++]=0){
             v=w        [i]+1;for(R                   =33;127  >R;R++)if(R-47&&R-92
             &&      R-(_)* w[i])*(                v++)=    (p)R;*v=0;}for(sprintf
                    /*'_ G*/  (*w+1,             "%0"    "8x",x(R=time(i=0),m,o)^~
                  0)  ;i<      8;++          i)u     [N+ i]=*(*w+i+1);for(*k=x(~
                    0,i=0    ,*a);i>-      1;     ){for (A=i;A<I;A++){u[+a [ A]
                  ]=w[A      ][e[A]] ;     k    [A+1]=x (k[A],a[A],a[A+1]
                ););}if    (R==k[I])     c(     (E(z[3  ],"wb+"))?fwrite(
               /* */  u,o,1,f)?D         ()|C(" \n   OK."):0   :C(
              "  \n  WriteError"         ))  for  (i  =+I-
             1  ;i >-1?!w[i][++            e[+ i]]:0;
               ) for( A=+i--;            A<I;e[A++]
               =0); (i <I-4              )?putchar
               ((_  )  46)                | fflush
              /*'     ,*/                ( stdout
              ):      0&                 0;}c(C
              ("      \n                fail")
             )     /*                  dP' /
                   dP                 pd  '
                  '                   zc
                                     */
                                     }
```

International Obfuscated C Code Contest (ioccc.org) - Don Yang 2004

Encryption/decryption tool.