Current Metrics

1. Efficiency
2. Operability
3. Effectiveness
4. Understandability
5. Learnability
6. Satisfaction
7. Attractiveness

Examples for requirements (Beavan 14, 15):

"all data entry clerks will be able to complete the task with at least 95% accuracy in under 10 minutes"

- ToT (uncontestable effort measure)
- Definition of user (experience level)
- Accuracy

"the mean score on the SUMI scale will be greater than 50" More information on quality in use requirements

Slide 4 (measuring usability)

Minimum effort – effort expended by an "expert" in completing a set of tasks.

Expected effort – effort expended by an experienced user

A "productive user" should expend effort at a level of <= 1.2 expert effort

The expert (level) Minimum effort level can be determined by measuring effort expanded by highly experienced users. Often, the developers can define the minimum level. Sometimes it can be mathematically inferred from curve (via the asymptote).

Emerson Control Studio - engineers design a system that simulates the control aspects of a process such as Oil Refinery. Used to enable (machine) control of the process. Alternatively enables a human operator to control the process.

– system A uses an old interface format of MS office. System B uses the Ribon a newer one

“Tool for measuring usability – learnability and operability - as a function of effort”:

System A. vs. System B. Usability Evaluation

- First, define a set of scenarios choose a subset.
- Next, for each scenario define a set of iid tasks (derived from the scenario with varying constraints.
- Determine the effort metrics (e.g., TOT, number of fixations, et.)
- Select a group of subjects for each system.
- Get the subjects to perform the tasks and measure the defined metrics.
- Use the average curve to identify learnability and operability (efficiency is highly correlated to operability)

System A. Vs. System B. is used for general assessment of Usability (WRT to tasks and apps)

e.g., TRACS vs. Canvas vs. blackboard (make vs. buy, buy-1 vs. buy-2, or continuous improvement)

Can we identify (pinpoint) usability issues in a way that directs the developers to the actual interface code.

In pinpoint analysis we segment the task (can be several minutes) into time segment units

1) Equal time (e.g., 30 seconds) uniform time segments
2) Event based a segment starts with an event (e.g., mouse click or KBD click) and ends with an event (mouse click, KBD click)

After segmentation, for each segment measure the selected metric[s] (e.g., number of fixations) for all the users executing this segment.

Decision function can be a threshold on the average value of the measured metric.

Training (with training subjects)

- Segment
- Measure the metric[s] per subject per segment
- Identify and set a threshold per segment (e.g., one standard deviation from the average of the result of measurements on a specific metric) for this segment.

Classification (E , NE) (with Classification subjects)

- Segment
- Measure the metric[s] per subject per segment
- Compare the average measurements to the threshold (per segment) of the average result with the training threshold. Above ➔ E (supervisor has to check), Below (NE – skip)

The recording of the entire sessions is several hours. How to find issues in the interaction?
Method 1) An expert evaluates the entire set of videos to identify issues.
Method 2 (pinpoint)) The expert only watches clips automatically identified as excessive effort clips.

Pros and cons
Method 2 pro reduces the time that the expert has to expand in watching video
Cons recognition errors

Type 1 - (False alarm) an NE segment is classified as E - increases expert time spent on watching interaction videos
Type 2 – False Negative an E is classified as NE - issues are not detected.

Possible "compromise" two pass; one which allows for higher error of one of the two types.
Second pass (tighter) to reduce the selected error.

There is an issue with including QTOpenGL in assignment I2.
Solution:
Add QT += opengl to the .pro file.

The assignments calls to add a graphics "window" to I1.
QT does not recognize GLU functions (glulookat(), gluperspective()) They can be used with the proper include (include glu.h in the right way).