Experiments

Spencer Smith
McMaster University, Canada
smiths@mcmaster.ca

Jacques Carette
McMaster University, Canada
carette@mcmaster.ca

Olu Owojaiye
McMaster University, Canada
owojaiyo@mcmaster.ca

Peter Michalski
McMaster University, Canada
michap@mcmaster.ca

Ao Dong
McMaster University, Canada
donga9@mcmaster.ca

1 User Experiments

1.1 Usability Experimental Procedure

1.2 Procedure

1. Survey participants to collect pre-experiment data
2. Participants perform tasks
3. Observe the study subjects (take notes, record sessions(OBS screen recorder), watch out for body languages and verbal cues)
4. Survey the study subjects to collect feedback (post experiment interview)
5. Prepare experiment report
6. Perform pairwise comparison analysis
7. Prepare analysis report

1.3 Task selection criteria

**The task selection will be determined with the aid of the domain experts attached to any of the selected projects.
**The domain experts will be asked to consider the below criteria when defining a task.
**Domain experts will also be asked to identify what background knowledge is necessary for the suggested tasks - Novice, Intermediate, Advanced

1. Collectively all tasks should not take no more than 2 hours.
2. Selected tasks should reflect common use cases of the software.
3. Include tasks that require a set of sequential or hierarchical steps to be completed

1.4 Usability Questionnaire

Two sources of standardized usability questionnaire we could use.
- https://www.usabilitest.com/sus-pdf-generator- 20-29 - SUS.
- https://uiuxtrend.com/pssuq-post-study-system-usability-questionnaire/ - PSSUQ

2 Modifiability Experiment

This experiment is designed to gather qualitative data regarding the modifiability of each software package in a domain. The steps outlined below will result in the creation, by domain experts, of a list of likely changes for software in the domain. This list will then be used to analyze which software packages have been designed to accommodate these expected modifications.

Parnas identifies likely changes as:

Likely Changes: if the system is required to be easy to change, the requirements should contain a definition of the areas that are considered likely to change. You cannot design a system so that everything is equally easy to change. Programmers should not have to decide which changes are most likelyParnas and Clements [1986]

2.1 Procedure

1. *Domain Expert*: List all likely changes that a domain expert might make in a software package in the domain.
2. Using the short list of software packages listed in the methodology document:
   a. Identify which of the above changes each package is likely designed to accommodate.
   b. Prepare a short report outlining the likely changes for software in the domain, and which changes each software package is likely designed to accommodate.

References

David Lorge Parnas and Paul C Clements. A rational design process: How and why to fake it. *IEEE transactions on software engineering*, (2):251–257, 1986.