# HCI Evaluation of Software

Beryl Plimmer, PhD  
 [beryl@cs.auckland.ac.nz](mailto:beryl@cs.auckland.ac.nz)

Computer software can be evaluated for its functional correctness and ease of use. There are many evaluation techniques that can be applied to new software tools. Which one to choose and what each will tell you is often confusing. We will survey the range of evaluation techniques from usability evaluations to cogitative dimensions and comparative studies. We will then look at comparative studies in detail.

Before we begin.

**Software Testing** is methodologies and techniques to verify the correctness of the software. The central question is ‘is the software functioning as expected?’

**HCI Evaluations** assume the software is functionally correct. With HCI evaluations the central question is ‘how well can people use the software?’ In many cases this is looking at one particular piece of software in isolation, effectively these are *usability evaluations*. However more interesting studies measure and compare specific interaction techniques or the efficacy (efficiency and effectiveness) of one piece of software over another. The following sections look first at usability evaluations and then *comparative studies*.

**Hint for thesis projects** One way to have a cohesive thesis is to start your requirements section with a persona and scenario (or two) - this gives your reader an idea of who might use your software. Then use the same persona(s) and scenario(s) for the evaluation study. I have indicated in the sections below where you might reuse the persona/scenario.

## Usability Evaluation

Usability evaluation is a check on the how easily the target audience can satisfy their goals. The assumption is that software is mostly used to complete some specific task – i.e. the user is goal oriented. For example, the user wants to send an email to a friend, or find what time flights leave for London.



???????

42

Usability evaluations can be categorized into two main types of evaluation: Those where usability experts review the software, and usability tests - those where potential users are involved in the evaluation.

## Expert Reviews

Expert reviews are evaluations of the software by an ‘expert’ this may be someone in the same organization, a knowledgeable friend or a specialist. The main categories of expert reviews are: heuristic evaluations, cognitive walk through, cognitive dimensions and guidelines review. The developer(s) cannot do a successful heuristic evaluation or cognitive walk-through on their own software. However it is possible to do cognitive dimensions and guidelines reviews on your own software.

### Heuristic evaluations

A heuristic evaluation is when an expert looks at an interface and comments on it in a general way. It can be conducted at any stage of the software development life-cycle. It could be conducted on a paper prototype, functional prototype or the final implemented system. The advantage of heuristic evaluations is that you only need one expert and one can expect that the expert would know what to look for. The disadvantage is that they are notoriously unreliable – some research suggests that experts find ~ 50% of the real errors and ~50% of the errors they identify are not real.

Most experts will employ techniques such as cognitive walkthroughs (see next section) or guidelines to conduct the evaluation. Guidelines such as Shneiderman’s eight golden rules [1] or Nielsen’s 10 Usability Heuristics (<http://www.useit.com/papers/heuristic/heuristic_list.html> ) are useful reminders for what to look for and prompts for structuring reporting. I have used some amalgam of these to write professional heuristic evaluation reports.

Shneiderman’s

1. *Strive for consistency*
2. *Enable frequent users to use shortcuts*
3. *Offer informative feedback*
4. *Design dialogs to yield closure*
5. *Offer error prevention and simple error handling*
6. *Permit easy reversal of actions*
7. *Support internal locus of control*
8. *Reduce short-term memory load*

Nielsen’s

1. *Visibility of system status*
2. *Match between system and the real world*
3. *User control and freedom*
4. *Consistency and standards*
5. *Error prevention*
6. *Recognition rather than recall*
7. *Flexibility and efficiency of use*
8. *Aesthetic and minimalist design*
9. *Help users recognize, diagnose, and recover from errors*
10. *Help and documentation*

In an academic setting you are most likely to get one or two lecturers or senior colleagues to do a heuristic evaluation for you. I would suggest that you give them a set of criteria such as those above and ask them to frame their comments to those criteria; this makes it to easier merge comments from different people. In addition give them your personas and scenarios so that they have a good feel for your intended users.

The validity of a heuristic evaluation depends to a large extent on the expertise of the evaluator. These types of evaluations are most often carried out when there is not enough time or money to do a usability test (with real users). A heuristic evaluation often flags false problems, things that the expert thinks ordinary users will find difficult, but in reality they don’t. The expert can also miss major real problems because of their familiarity with software tools in general. One strategy you could employ is to blend a heuristic evaluation with some other type of evaluation.

### Cognitive walk through

A cognitive walk through is similar in approach to a heuristic evaluation [2]. However, the technique has its origin in code walkthroughs. The evaluators step through all the actions that are required to perform a particular task. Below is a complete description of what and how to do a cognitive walk through from <http://www.cc.gatech.edu/computing/classes/cs3302/documents/cog.walk.html> .

|  |
| --- |
| CS 3302 Introduction to Software Engineering Winter 1995 Gregory AbowdPerforming a Cognitive Walkthrough This explanation of the cognitive walkthrough is adapted from the following sources:  Alan Dix, Janet Finlay, Gregory Abowd and Russell Beale, *Human-Computer Interaction*, Prentice Hall, International, 1993. Chapter 11 contains information on evaluation techniques.  Clayton Lewis and John Rieman, *Task-Centered User Interface Design: A practical introduction.* A shareware book published by the authors, 1993. Original files for the book are available by FTP from ftp.cs.colorado.edu.  Cathleen Wharton, John Rieman, Clayton Lewis and Peter Polson, *The Cognitive Walkthrough: A practitioner's guide*. In Jakob Nielsen and Robert L. Mack, editors, *Usability Inspection Methods*. John Wiley and Sons, Inc. 1994.  The origin of the cognitive walkthrough approach to evaluation is the code walkthrough familiar in software engineering. Walkthroughs require a detailed review of a sequence of actions. In the code walkthrough, the sequence represents a segment of the program code that is stepped through by the reviewers to check certain characteristics (e.g., that coding style is adhered to, conventions for spelling variables versus procedure calls, and to check that system wide invariants are not violated).  In the cognitive walkthrough, the sequence of actions refers to the steps that an interface will require a user to perform in order to accomplish some task. The evaluators then step through that action sequence to check it for potential usability problems. Usually, the main focus of the cognitive walkthrough is to establish how easy a system is to learn. More specifically, the focus is on learning through exploration. Experience shows that many users prefer to learn how to use a system by exploring its functionality hands on, and not after sufficient training or examination of a user's manual. So the kinds of checks that are made during the walkthrough ask questions that address this exploratory kind of learning. To do this, the evaluators go through each step in the task and provide a story about why that step is or is not good for a new user.  To do a walkthrough (the term walkthrough from now on refers to the cognitive walkthrough, and not any other kinds of walkthroughs), you need four things.   1. A description of the prototype of the system. It doesn't have to be complete, but it should be fairly detailed. Details such as the location and wording for a menu can make a big difference. 2. A description of the task the user is to perform on the system. This should be a representative task that most users will want to do. **\*** 3. A complete, written list of the actions needed to complete the task with the given prototype. 4. An indication of who the users are and what kind of experience and knowledge the evaluators can assume about them. **\***   Given this information, the evaluators step through the action sequence (item 3 above) to critique the system and tell a believable story about its usability. To do this, for each action, the evaluators try to answer the following four questions.  **A. Will the users be trying to produce whatever effect the action has?**  Are the assumptions about what task the action is supporting correct given the user's experience and knowledge up to this point in the interaction?  **B. Will users be able to notice that the correct action is available?**  Will users see the button or menu item, for example, that is how the next action is actually achieved by the system? This is not asking whether they will know that the button is the one they want. This is merely asking whether it is visible to them at the time when they will need to invoke it. An example of when this question gets a negative supporting story might be if a VCR remote control has a hidden panel of buttons that are not obvious to a new user.  **C. Once users find the correct action at the interface, will they know that it is the right one for the effect they are trying to produce?**  This complements the previous question. It is one thing for a button or menu item to be visible, but will the user's know that it is the one they are looking for to complete their task?  **D. After the action is taken, will users understand the feedback they get?**  Assuming the users did the correct action, will they know that. This is the completion of the execution/evaluation interaction cycle. In order to determine if they have accomplished their goal, the user needs appropriate feedback. Important Note It is vital to document the cognitive walkthrough to keep a record of what is good and what needs improvement in the design. It is therefore good to produce some standard evaluation forms for the walkthrough. The cover form would list the information in items 1-4 above, as well as identify the date and time of the walkthrough and the names of the evaluators. Then for each action (from item 3 on the cover form), a separate standard form is filled out that answers each of the questions A-D above. Any negative answer for any of the questions for any particular action should be documented on a separate usability problem report sheet. This problem report sheet should indicate the system being built (the version, if necessary), the date, the evaluators and a detailed description of the usability problem. It would also be useful to determine the severity of the problem, that is, whether the evaluators think this problem will occur often and an impression of how serious the problem will be for the users. This information will help the designers to decide priorities for correcting the design. |

\* reuse your personas and scenarios for the cognitive walkthroughs.

### Guidelines review

A guidelines review is checking the user interaction against a well defined set of criteria. The most common guidelines are the w3c web accessibility guidelines for visually impaired (<http://www.w3.org/TR/WCAG20/>). There are some specific points in these guidelines for tagging images and such, however most of the guidelines are simply good design guidelines. You will notice quite a lot of commonality between the words on the table of contents below and the guidelines listed under heuristic evaluations.

|  |
| --- |
| WCAG 2.0 Guidelines   1. Perceivable    1. Provide text alternatives for any non-text content so that it can be changed into other forms people need, such as large print, braille, speech, symbols or simpler language    2. Provide synchronized alternatives for synchronized media    3. Create content that can be presented in different ways (for example simpler layout ) without losing information or structure    4. Make it easier for users to see and hear content including separating foreground from background 2. Operable    1. Make all functionality available from a keyboard    2. Provide users with disabilities enough time to read and use content    3. Do not design content in a way that is known to cause seizures    4. Provide ways to help users with disabilities navigate, find content and determine where they are 3. Understandable    1. Make text content readable and understandable    2. Make Web pages appear and operate in predictable ways    3. Help users avoid and correct mistakes 4. Robust    1. Maximize compatibility with current and future user agents, including assistive technologies |

The difference between these guidelines and the heuristics is that they are very specific and measurable. The guidelines also have explanations and measures of how to meet them. In the box below is one example (from <http://www.w3.org/TR/WCAG20/#keyboard-operation>).

|  |
| --- |
| Principle 2: Operable - User interface components and navigation must be operable  Guideline 2.1 Keyboard Accessible: Make all functionality available from a keyboard Understanding Guideline 2.1  2.1.1 Keyboard: All functionality of the content is operable through a keyboard interface without requiring specific timings for individual keystrokes, except where the underlying function requires input that depends on the path of the user's movement and not just the endpoints. (Level A) How to Meet 2.1.1 Understanding 2.1.1  Note 1: This exception relates to the underlying function, not the input technique. For example, if using handwriting to enter text, the input technique (handwriting) requires path dependent input but the underlying function (text input) does not.  Note 2: This does not forbid and should not discourage providing mouse input or other input methods in addition to keyboard operation. |

It is possible to check your own interface against these guidelines. There are also some online checkers for web pages. Bobby <http://webxact.watchfire.com/> is the most well know. It is extremely simple to use (have a try). From a research perspective it is probably a useful thing to report, particularly if your research is about web stuff (eg automatic generation of webpages or web-base learning tools). There are a number of other research initiatives for automatic testing of user interfaces, but I am not aware of any that are available publically.

### Cognitive Dimensions

Cognitive dimensions is a framework for analyzing the usability of software. Green and Blackwell (<http://www.cl.cam.ac.uk/~afb21/CognitiveDimensions/>) developed this technique from their joint psychology and computer science background. It’s a more thorough approach than either heuristic evaluations or cognitive walkthroughs. You can do a cognitive dimensions evaluation on your own software – however it does require quite a bit of thought and reflection on your part to do it well.

***Main Cognitive Dimensions***

(from [***http://www.cl.cam.ac.uk/~afb21/CognitiveDimensions/CDtutorial.pdf***](http://www.cl.cam.ac.uk/~afb21/CognitiveDimensions/CDtutorial.pdf) )

* Abstraction types and availability of abstraction mechanisms
* Hidden dependencies important links between entities are not visible
* Premature commitment constraints on the order of doing things
* Secondary notation extra information in means other than formal syntax
* Viscosity resistance to change
* Visibility ability to view components easily
* Closeness of mapping closeness of representation to domain
* Consistency similar semantics are expressed in similar syntactic forms
* Diffuseness verbosity of language
* Error-proneness notation invites mistakes
* Hard mental operations high demand on cognitive resources
* Progressive evaluation work-to-date can be checked at any time
* Provisionality degree of commitment to actions or marks
* Role-expressiveness the purpose of a component is readily inferred

You will note that there are many of the same ideas as in the previous types of evaluations. Below is a copy of the explanation of viscosity from the on-line tutorial.

|  |
| --- |
| **Viscosity**  **Definition**  Resistance to change: the cost of making small changes.  *Repetition* viscosity: a single goal-related operation on the information structure (one change 'in the  head') requires an undue number of individual actions,.  *Knock-on* viscosity: one change 'in the head' entails further actions to restore consistency  **Thumbnail illustrations**  Repetition viscosity: Manually changing US spelling to UK spelling throughout a long document  Knock-On viscosity: inserting a new figure into a document creates a need to update all later figure  numbers, plus their cross-references within the text; also the list of figures and the index  **Explanation**  Viscosity in hydrodynamics means resistance to local change – deforming a viscous substance, e.g.  syrup, is harder than deforming a fluid one, e.g. water. It means the same in this context. As a working  definition, a viscous system is one where a single goal-related operation on the information structure  requires an undue number of individual actions, possibly not goal-related. The ‘goal-related operation’  is, of course, at the level that is appropriate for the notation we are considering. Viscosity becomes a  problem in opportunistic planning when the user/planner changes the plan; so it is changes at the plan  level that we must consider. This is important, since changes at a lower level can sometimes be pure  detail (inserting a comma, for instance), giving a misleading impression of low viscosity. So, in general,  *viscosity is a function of the work required to add, to remove, or to replace a plan-level component of an information structure.*  Note that viscosity is most definitely a property of the system as a whole. Certain notations give the  potential for viscosity (e.g. figure numbers) but the environment may contain tools that alleviate it by  allowing aggregate operations (e.g. a smart word-processor with auto-number features). See below for  associated trade-offs.  Also note that viscosity may be very different for different operations, making it hard to give a  meaningful overall figure.  **Cognitive relevance**  Breaks train of thought, gets too costly to change or to take risks; encourages reflection and planning,  reduces potential for disastrous slips and mistakes.Acceptable for transcription activity and for simple incrementation activity, but problematic for exploratory design and for modification activity.  **Cost Implications**  It is easy for designers to view a device as a pure incrementation system and overlook the need for  restructuring. *This is a serious and far too frequent mistake.* So a computer-based filing system may readily  allow new categories to be added, but have very little provision for redefining categories and recategorising  the objects within them.  The worst problems come when viscosity is combined with premature commitment. The user has to  make an early guess (that’s the premature commitment) but then finds that correcting that guess is  time-consuming or costly (that’s the viscosity).  **Types and Examples**  *Repetition viscosity* is a frequent problem in structures which exist in the user’s mind rather than being  recognised by the system. A **collection of files** making up one document in the user’s mind may need  to be edited to bring their typography into conformance, usually by editing each file individually,  because few systems have the ability to apply the same editing operations to each file in a list.  *Knock-on viscosity* is the other main type, frequently found in structures that have high interdependencies,  such as **timetables**:    In practice, potent combinations of knock-on and repetition viscosity are the type to be feared, turning  up frequently in **graphics structures**. Although word-processors are steadily reducing the viscosity  problem for document generation, drawing packages are not making comparable progress. Editing a  drawing usually requires much tedious work, and frequently many similar alterations need to be made  to different parts of the picture; automation tools, desirable as they might be, are not yet commercially  available.    Figure 4 This tree-chart, showing part of the JavaScript object hierarchy, illustrates  both knock-on viscosity and potential repetition viscosity: if the ‘window’ box is moved  to one side, all the lines connecting it to other boxes will have to be moved (knockon).  In most drawing systems each line will have to be redrawn individually (repetition  viscosity).  Drawing a **genealogical tree** for any reasonably extensive family will well illustrate the difficulties.  Many **programming languages** exhibit pronounced viscosity. This is a serious problem, because  software is always changing – during development and also after it is supposed to be finished. Many of  the trends in programming language development have therefore been aimed at reducing viscosity.  One of the most recent is object-oriented programming, which allows a single change to be made in a  parent class, thereby altering the behaviour of many different subclasses. Unfortunately this has the  side effect of introducing a new kind of viscosity - if the interface to a class changes, rather than its  behaviour, this often results in knock-on changes to its parent class, as well as to all its siblings, and the  contexts in which they are invoked (even though they may be only tenuously related to the original  change).  So far we have described non-interactive examples, but the viscosity concept is readily extended to  interactive systems. Think of using devices as uttering sentences in an interaction language. For noninteractive  devices (i.e. those where the names of actions persist and their execution is delayed, such as  programs and sheet music) viscosity obviously consists in the work required to reshape the actions. For  interactive devices, in which the actions are executed as soon as named (e.g. dialling a telephone,  finding a given radio programme) viscosity consists in the work required to get from one state of the  device to another state: for the **telephone**, after a slip of the hand has caused a wrong digit to be dialled,  the only remedy is to cancel the transaction and restart; for the radio, changing to a different  programme may be easy or difficult (see example below).  ***Other contexts***  **Hypertext structures**, although easy to create, can become extremely viscous. Fischer (1988), describing  what appears to be the process of opportunistic design that takes place while creating a collaborativelygenerated  hypertext structure, writes “ ... For many interesting areas, a structure is not given a priori  but evolves dynamically. Because little is known at the beginning, there is almost a constant need for  restructuring. [But] despite the fact that in many cases users could think of better structures, they stick  to inadequate structures, because the effort to change existing structures is too large.”  **Tailorable systems** seem to create opportunities for viscosity at the organizational level. Examples of  these systems include customisable settings and macros for spreadsheets and word-processors, and  other systems allowing users to tailor their office environment. The problem here is that users lose the  ability to exchange tools and working environments; moreover, the arrival of a new release of the  software may mean that it has to be tailored all over again, possibly with few records of what had been  done before.  Indeed, many organizations are repositories of information among many people (‘distributed  cognition’) and the organizations themselves can be therefore be considered as information structures.  Changing a component of the information held collectively by the organisation may be a matter of  telling every single person separately (Repetition Viscosity), or it may, in a hierarchical or bureaucratic  organization, only need one person to be told. The levels of the hierarchy have the same role as the  abstractions in other systems.  ***Viscosity growth over time***  Finally, diachronic aspects of viscosity. In many **design systems**, the viscosity of any one structure or  design tends to increase as it grows. In the context of CAD-E design, one workplace has reported that  their practice is to explicitly declare a point after which a design can no longer be changed in any  substantial way. At that point the design is said to have ‘congealed’, in their terminology, and  opportunistic design must be restricted to details. There is at present no reported data, to my  knowledge, describing in detail the time course of viscosity in different systems. The impression is that  certain programming languages, such as Basic, have low initial viscosity, but as programs grow their  viscosity increases steeply. Pascal starts higher but increases more slowly, because more is done in the  initial abstractions. Object-oriented systems are intended to preserve fluidity even better, by using late  binding and the inheritance mechanism to minimise the effort needed to modify a program. Some of  the investigations made under the banner of software lifecycle studies may prove to be revealing.  **Workarounds, Remedies and Trade-offs**  Before considering possible remedies, observe that viscosity is not always harmful. Reduced viscosity  may encourage hacking; increased viscosity may encourage deeper reflection and better learning.  Reduced viscosity may allow small slips to wreak disaster; for safety-critical systems, high viscosity  may be more desirable.  Nevertheless, viscosity is usually to be avoided, and some of the possible manoeuvres are:  $ *The user decouples from the system*: the workaround for unbearable viscosity (especially when combined  with a need for early or premature commitment) is to introduce a different medium with lower  viscosity to make a draft version, and then transfer it to the original target medium. Thus, the process is  now two-stage: an *exploratory* stage followed by a *transcription* stage.  Thus, before drawing in ink, an artist may make a sketch in pencil, which can be erased. A programmer  may work out a solution on paper before typing it.  Alternatively, in some circumstances users may decouple by adopting a different notation instead of a  different medium. The programmer may choose to develop a solution in a ‘program design language’  before transcribing it into the target language.  $ *Introduce a new abstraction*. This is the classic remedy. Got a problem with updating figure numbers? OK,  we’ll create an AutoNumber feature. See below for the trade-off costs of abstractions.  $ *Change the notation*, usually by shifting to a relative information structure instead of an absolute one.  Consider a list of quick-dial codes for a **mobile telephone**:    Figure 5 Modifying the order of entries in a telephone memory is usually a high viscosity operation  I want to be able to dial ‘Car rescue service’ very quickly indeed, let us say. That means moving it to the  top of the list, because on this model of mobile phone the dialling codes are scrolled one by one until  the required target is visible. I want to move that one item while keeping the others in their existing  order. Almost certainly, on existing models, I will have to re-enter the whole list; but in a good **outliner**  system, I could perform exactly the same modification in a single action, by selecting the Car rescue  entry and dragging it upwards:    Figure 6 Order modification in an outliner is a low-viscosity operation. The highlighted  item (car rescue service) is being dragged upwards; when released, it will take its  place where the horizontal arrow points, and all lower items will be displaced  downwards.  Many other dimensions can affect viscosity, but it will obviously be easier to discuss such trade-offs  when we reach the dimensions concerned. |

Similar descriptions are available in the tutorial for the first 6 dimensions.

A student of mine, Richard Priest used cogitative dimensions to evaluate his Master’s thesis. A copy of his thesis is available on my web site ([http://www.cs.auckland.ac.nz/~beryl/publications/MSc Richard Priest 2006 Thesis.pdf](http://www.cs.auckland.ac.nz/~beryl/publications/MSc%20Richard%20Priest%202006%20Thesis.pdf) ).

One of the advantages of this approach is you can do it by yourself and therefore you do not need ethical approval. If you decide on this approach, don’t be too hard on yourself!

# User Studies

The essential difference between expert reviews and user studies is that with user studies *potential or real users* are asked to operate and comment on the software. This is a user-centred design approach to software evaluation.

### Tips and techniques

* Most (if not all) universities and many organizations require you to get ethics approval before you can do a usability study. It often takes months so start planning early.
* Record everything – use a product like Morae (<http://www.techsmith.com/> )if at all possible. Try to include the right to use digital recordings and pictures in your ethics approval – they are very handy for pics in your publications and videos for presentations. Sometimes you may have to fuzzy out faces, but that isn’t hard.
* Pilot, pilot and pilot.
* Don’t be afraid to fix big problems after the pilot – or half way through the study.

## Classic Usability Testing

With a class usability test you ask potential users to try to complete specific tasks with the software. Before they begin you will probably want to gather some demographic and current skill level information from them. While they are doing the tasks you observe and take notes. Afterwards you get them to rate/comment on the experience.

The key to a good usability study is very careful planning. Here is the planning document I use with my students.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Usability Testing Template**  **Project Name:**  **Name of Tester(s):**  **Supervisor:**  **Ethics Approval Number:**  **Overview:** *one or two sentences describing the project goals*  **Usability Test Type:**  *Define the type of test:*  *Exploratory – establish the intuitiveness of the implementation*  *Assessment – measure the effectiveness of the implementation*  *Validation – establish the product performance*  *Comparison – compare pros/cons of different designs*  **Purpose:**  *Define the purpose of the usability study.*  **Test Environment / Equipment:**  *List the type of equipment is needed to conduct the usability test (computer, video capture software, video cameras)*  *Describe the environment where the testing will take place.*  *Be sure that the recording is covered under the ethics approval*  **Test Objectives:**  *Define the functions of your application that are to be tested.- usually this will be the main use-cases*  **Profile:**  *Define the type of users suitable for the participation of your software. Your software may require participants that are knowledgeable/specialized in some fields (knows how to use an IDE).*  *If you have defined some personas for your project match the profile to these.*  **Task Design:**   * *A task is used to test out a particular area of the software.* * *Each task is based on a scenario. You may require many scenarios to test out different parts of the system.* * *Scripted tasks are recommended as the results will be more consistent and comparable if the user is guided.* * *Each participant uses the SAME scenario/example to keep consistency.* * *You may want to preload samples in order to test a particular functions.*   **Pilot Testing:**  *This is a dummy run of your usability test. This is to test out your usability questions to determine if they are suitable (Question/Tasks may be too difficult/easy/long/short) and to make changes if they aren’t.*  **Acquiring Participants:**  *Can be done via email/invitation notices/word of mouth. – check the ethics*  *You need to determine the number of participants required for your test - 8 is usually enough – my rule of thumb is when you haven’t found any new problems with the previous two participants you have probably done enough.*  **Actual Test**  **Test Preparation (Training):**  *- read participant information sheet*  *- sign ethics approval*  *- Pre-test questionnaire.*  *- User Training*  *- Explain what the project is about and what the software does (background knowledge)*  *- Getting the user familiar with the equipment*  *- Demonstrate the functions of the software with an example.*  *- You may want to train the user in sections so as to not overload the user with excessive information*  **Actual Testing:**  *Check the recording devices are turned on*  *Ask participants to complete particular tasks.*  **Post Test**  *- Post-test questionnaire / informal interview*  *- This is where qualitative information is acquired such as how easy it is to use the software, how easy the tasks were.*  *- The questionnaire may also be filled out after each scenario.*  *- Thank the participant – hand out reward/thank you*  **Evaluation Analysis:**  *The measures that will be used to evaluate the application. Examples of such measures are as follows:*  *Quantitative:*  *Time taken for each tasks Mistakes made, Recognition rate*  *Qualitative:*  *User emotions during test (ie frustration/irritation), User feedback from questionnaire.*  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Sample Advertisement/ email/notice    Department of Computer Science  The University of Auckland  Private Bag 92019  Auckland  Tel: 09 373 7599  My name is Beryl Plimmer, I am a Senior Lecturer in the Department of Computer Science at the University of Auckland. I am conducting research into **{pen | gesture}** input computing **{with *name* RA | project student}** working under my supervision. I am investigating how computers can support more natural computer interaction. A part of exploring these ideas is involving potential ‘ordinary’ users in the design, usability testing and evaluation of the prototype applications. This particular project aims to **{ eg support hand-drawing of *diagram type* | provide hand gesture interaction to 3D environments | support digital ink annotation in software development tools | other as appropriate }**  In this study we are **{collecting base data for training our recognition engines | asking people how they conduct particular tasks now and how they think these could be translated into input gestures | testing the usability of a prototype system for *application* | comparing a new prototype for *application* to other tools for completing the same task}.** Thestudies are conducted in the HCI Lab in Computer Science and will take a maximum of **{30 | 60 | 2 hours}**  You are invited to participate in our research and we would appreciate any assistance you can offer us, although you are under no obligation to do so. The studies will take place between *date and date*. If you would like to participate please email|phone **{me | RA | project student}** to arrange a time a*ppropriate contact info*  Regards  Beryl Plimmer  This research has been approved by ………………………………..   |  |  | | --- | --- | |  | Department of Computer Science  The University of Auckland  Private Bag 92019  Auckland  Tel: 09 373 7599 |   **PARTICIPANT INFORMATION SHEET**  **Title: Gesture input for computer applications**  To participants:  My name is Beryl Plimmer, I am a Senior Lecturer in the Department of Computer Science at the University of Auckland. I am conducting research into **{pen | gesture}** input computing **{with *name* RA | project student}**. I am investigating how computers can support more natural computer interaction. A part of exploring these ideas is involving potential ‘ordinary’ users in the design, usability testing and evaluation of the prototype applications. This particular project aims to **{ eg support hand-drawing of *diagram type* | provide hand gesture interaction to 3D environments | support digital ink annotation in software development tools | other as appropriate }**  In this study we are **{collecting base data for training our recognition engines| asking people how they conduct particular tasks now and how they think these could be translated into input gestures| testing the usability of a prototype system for *application* | comparing a new prototype for *application* to other tools for completing the same task}.**  You are invited to participate in our research and we would appreciate any assistance you can offer us, although you are under no obligation to do so.  Participation involves one visit to our laboratory at The University of Auckland, for approximately {3**0 | 60 | 2 hours**}, and requires that your eyesight is normal or corrected-to-normal by spectacles or contact lenses. If you agree to participate, you may be asked to perform a number of tasks on paper, using a computer via the keyboard or mouse and using a computer with a pen. The tasks will be fully explained and demonstrated. You will be asked to **{{construct | review | correct} *type* diagrams| annotate a documents| review annotated documents| interact with a virtual 3D environment}***.* The changes you make and the time you spend working on each task will be digitally recorded together with synchronized video. You will be asked fill in a short questionnaire to note your age, education level and existing experience with the tasks and technology and complete a short questionnaire on your experience.}  All the questionnaire information you provide will remain anonymous. The digital recordings, with your specific consent, may be used in research reports on this project. You choose whether your recordings are used or not on the consent form. Your consent form will be held in a secure file for 6 years, at the end of this time it will be properly disposed of. Your name will not be used in any reports arising from this study. The information collected during this study may be used in future analysis and publications and will be kept indefinitely. When it is no longer required all copies of the data will be destroyed. At the conclusion of the study, a summary of the findings will be available from the researchers upon request.  If you don’t want to participate, you don’t have to give any reason for your decision. If you do participate, you may withdraw at any time during the session and you can also ask for the information you have provided to be withdrawn at any time until one week after the conclusion of your session, without explanation and without penalty, by contacting me (details below). If you are a student at The University of Auckland choosing not to participate, or to withdraw yourself or your information, your grades or academic relationships with the University or members of staff will not be affected.  If you agree to participate in this study, please first complete the consent form attached to this information sheet. Your consent form will be kept separately from your questionnaire data so that no-one will be able to identify your answers from the information you provide.  This project is partly supported by funding from Microsoft Research Asia and University of Auckland Faculty of Science Research Fund project number 360933/9343  Thank you very much for your time and help in making this study possible. If you have any questions at any time you can phone me (3737599 ext 82285) or the Head of Department, Associate Professor Robert Amor (3737599 ext 83068), or you can write to us at:  Department of Computer Science,  The University of Auckland  Private Bag 92019  Auckland.  For any queries regarding ethical concerns, please contact ………………….  APPROVED BY ………………… .   |  |  | | --- | --- | |  | Department of Computer Science  The University of Auckland  Private Bag 92019  Auckland  Tel: 09 373 7599 |   **CONSENT FORM**  **This consent form will be held for a period of at least six years**  **Title: Investigation of gesture input computing**  **Researcher:** Dr Beryl Plimmer  I have been given and understood an explanation of this research project. I have had an opportunity to ask questions and have them answered. I understand that at the conclusion of the study, a summary of the findings will be available from the researchers upon request.  I understand that the data collected from the study will be held indefinitely and may be used in future analysis.  I understand that I may withdraw myself and any information traceable to me at any time up to one week after the completion of this session without giving a reason, and without any penalty.  I understand that I may withdraw my participation during the laboratory session at any time.  I understand that my grades and relationships with The University of Auckland will be unaffected whether or not I participate in this study or withdraw my participation during it.  I agree to take part in this research by completing the laboratory session.  I confirm that my eyesight is normal or corrected-to-normal.  I **agree/do not agree** digital and video recordings taken during the session being used research reports on this project.  Signed: Name:    Date:  APPROVED BY ………………………. .  **Sample Questionnaire – to be tailored to particular tasks**   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **Complete only This Section before the session** |  |  |  | |  | |  |  | | | I use a xxx for xxx task |  | Always | | Usually | | Sometimes | | | Rarely | | Never | | I use a xxx for xxx tasks |  | Always | | Usually | | Sometimes | | | Rarely | | Never | | I use computer tools for xxxx tasks |  | Always | | Usually | | Sometimes | | | Rarely | | Never | | I have used a xxxx input on a computer |  | Frequently | | Occasionally | | A couple of times | | | Once | | Never |   **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **The following sections will be completed in two parts**  **Complete this section after the Complete this section after the  first exercise second exercise   Task \_\_\_\_\_\_\_\_\_\_\_\_** **Task \_\_\_\_\_\_\_\_\_\_\_\_\_**  **Tools \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Tools \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**   |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  | **Strongly Agree** | **Agree** | **Neutral** | **Disagree** | **Strongly Disagree** |  | **Strongly Agree** | **Agree** | **Neutral** | **Disagree** | **Strongly Disagree** | | **General Questions** |  |  |  |  |  |  |  |  |  |  |  |  | | This exercise was enjoyable |  |  |  |  |  |  |  |  |  |  |  |  | | **About the task** |  |  |  |  |  |  |  |  |  |  |  |  | | I understand the task |  |  |  |  |  |  |  |  |  |  |  |  | | This interaction tools helped with my task completion |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  | | **About the environment** |  |  |  |  |  |  |  |  |  |  |  |  | | Creating the sketch/diagram/annotations/ was easy |  |  |  |  |  |  |  |  |  |  |  |  | | Checking and editing the sketch/diagram/annotations/ was easy |  |  |  |  |  |  |  |  |  |  |  |  | | I would like to use this method of interaction in the future |  |  |  |  |  |  |  |  |  |  |  |  |   Given a choice my preference for completing *task* would be  Tool 1 🞎 Tool 2 🞎 None 🞎  Comments/Recommendations:   ---------------------------------------------------------------------------------------------------------------------------------  ---------------------------------------------------------------------------------------------------------------------------------  ---------------------------------------------------------------------------------------------------------------------------------  --------------------------------------------------------------------------------------------------------------------------------- |

If you are using Morae you can do the questionnaire in Morae – it has standard questions similar to those above. You can also add your own questions. And it does the basic questionnaire analysis for you (how cool is that!). One big advantage of doing a plan like this is that most of the plan can cut and paste fairly directly into your thesis.

A classic usability study gives you a fairly good idea of how usable your system is. Be warned you **will** find ‘really dumb’ problems that you should have been able to predict! Remember, be kind to yourself. Get some advice and get in and fix the worst problems – it doesn’t usually take very long.

Mobile devices require special planning. The context of use has quite an effect on its usability. Doing something in a quite lab with no distractions is not the same as doing it in a lecture theatre with 100 students watching you make an idiot of yourself.

Other projects may require you to think outside the square. I have been working on a haptic interface to teach blind kids how to sign their name. There are very few blind kids so we asked blind adults to do the usability testing for us. This project is to be presented soon at Chi 2008. A copy is on my web page (<http://www.cs.auckland.ac.nz/~beryl/publications>).

Whatever your project there are probably some papers presenting usability studies on similar tools, have a search around the web. The New Zealand ‘expert’ on how to run and report on usability studies is Andy Cockburn, Canterbury. Have a look at some of his papers if you want some good examples of studies and write-ups. (ACM author search “Cockburn, A” uncovers a few).

## Comparative Evaluations

Comparative evaluations are the most interesting and the most difficult to plan and execute well. With a comparative evaluation you ask people to complete similar tasks with two (or more) different tools. What you compare, will depend on what your project is focused on. If you have developed a super cool new fisheye interface for phones then you might want compare it to a standard display or some other fish-eye. With my work I am often comparing doing the task on (real) paper to doing it with a sketch tool, to doing it with a widget based tool. Each thing that you are comparing is called a ‘condition’.

You can use the planning template above, adding the following.

What **exactly** are you comparing?

Popular things are: time, error rates, user satisfaction. However depending on your project there are all sorts of other measures. I am planning a comparison between two mind-mapping tools, a widget based tool and a sketch tool. The main point of mind mapping is generating lots of interrelated ideas. We can count the number of ideas generated in each tool (conveniently we found a paper that talks about success in terms of number of ideas generated). However, we have to be sure that we keep everything else as similar as possible.

**Latin squares:** you need to give each condition an equal chance of success. That means you need as many tasks as conditions. Each task must be as similar as you can possibly make it, without being the same. You then need to vary the order of condition/tasks. For a simple two way comparison you have four combinations

|  |  |  |
| --- | --- | --- |
|  | Condition A | Condition B |
| Task 1 | A1 | B1 |
| Task 2 | A2 | B2 |

However when you vary the order you have eight combinations

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Order 1 |  |  | Order 2 |  | |
|  | Condition A | Condition B |  | Condition B | | Condition A |
| Task 1 | A1 | B1 |  | B1 | A1 | |
| Task 2 | A2 | B2 |  | B2 | A2 | |

You can see that increasing the number of conditions increases the number of combinations very quickly!

**Equal Tasks** making up tasks of equal difficulty, but not the same is really hard! Depending on the tool and problem space you can sometimes pick the tasks up from textbook exercises or practice exam questions. Otherwise you have to try to balance the tasks by measure the difficulty, number of attributes, number of steps, or similar. You should statistically analyze the results to see if there is a difference in performance between tasks.

**Plan your statistics.** If you have done a good few statistics papers you can possibly do this yourself. I am not a statistician so I often go and ask for advice. There are a scary number of different statistical measures you can do on this type of data. Excel probably won’t cut it for this type of statistical analysis. SPSS is probably a better choice. ANOVAs are probably a basic starting point. I have found these two books useful references

Paul T. Kinnear, Colin D. Gray, SPSS 14 Made Simple, Psychology Press, 2006

George A. Ferguson, Yoshio Takane, Statistical Analysis in Psychology and Education, 6th Edition, McGraw-Hill, 1989

**Extend your Pilot testing**  this type of evaluation is a major undertaking. You want to be sure that you have good tasks, good questions, are collecting the right data and can analyze it! Plan to pilot test with 3-4 people all the way through the study – including the statistical analysis.

A two-way evaluation would certainly be sufficient for a CS Masters project. Psychology students often do much more complex evaluations – but then the evaluation *is* the project, rather than verification. Louise Yeung did a 5 way evaluation of visual fidelity for her Psychology Masters (<http://www.cs.auckland.ac.nz/~beryl/publications/> .. Louise Yeung Thesis 2006.doc ). I wouldn’t recommend an evaluation like this for a CS major, but a look at her thesis and the related papers shows the interesting things you can discover. My PhD was a more modest three way evaluation ([http://www.cs.auckland.ac.nz/~beryl/publications/PhD Thesis Using Shared Displays to Support Group Design b.pdf](http://www.cs.auckland.ac.nz/~beryl/publications/PhD%20Thesis%20Using%20Shared%20Displays%20to%20Support%20Group%20Design%20b.pdf) ).

# Summary

If you are doing a software engineering, HCI or graphics project at Hons, Masters or PhD level that includes developing software your supervisor will probably recommend that you include an evaluation study. What type of evaluation you choose to do will depend on

* The size/level of your project (some Hons projects do not have and evaluation – some PhDs have two or three)
* How much time you have
* What resources you have available
* Whether you can get appropriate ethics approval (in the time available)
* What your supervisor recommends
* What you feel like doing

Basing your evaluations on one of the standard methodologies and making sure you follow the methodology validates your work. Spend time reading up about the methodology and look for published papers that use the methodology of choice – they will give you ideas on what to look for and how to present your results.

You, can of course, compare evaluation methodologies as a project!

Happy Evaluating

Beryl Plimmer PhD

[beryl@cs.auckland.ac.nz](mailto:beryl@cs.auckland.ac.nz), University of Auckland

# References

1 Shneiderman, B., Plaisant, C., Designing the user interface, Pearson Addison Welsey, (2005),

2 Wharton, C., Rieman, J., Lewis, C., Peter Polson. In , e., The Cognitive Walkthrough: A practitioner's guide, Usability Inspection Methods, John Wiley and Sons, Inc., (1994),