Usability, user-centered design (UCD) and FOSS OSDC conference 2006

Author: Scott Rippon, Monash University (http://www.monash.edu/)

Abstract: This paper provides an introduction to usability and user-centered design (UCD), the benefits of it's use, and details on a some common UCD techniques. Some present usability initiatives within the FOSS community are highlighted, finishing with some discussion of the challenges facing further adoption of UCD.

What is user-centred design (UCD) and usability?

User-centred design is an design approach/philosophy where users are involved in the planning, design and development phases in a systems development. The goal of performing UCD, and focusing on users, is to ensures that the system developed is usable. UCD is an iterative process where we design, evaluate and then repeat.

Whilst UCD is relatively easy concept, usability can be difficult. The term “usability” can be used in number of different contexts which produce different meanings. For example it can be used to describe:

- an **outcome**;
  eg. we've increased the usability of this web site.
- a **process**; or
  eg. let's use a usability approach when developing this system.
- a **set of techniques**
  eg. there are a number of usability techniques that we can perform (eg. usability testing, card sorting, contextual inquiry, etc.) to improve ease of use of the systems.

There many many different available definitions. They range from the very broad, where usability refers to improving the ease-of-use, to the specific:

> [Usability refers to] the extent to which a product can be used by specified users to **achieve specified goals** with **effectiveness**, **efficiency** and **satisfaction** in a specified context of use. (ISO 9241-11)

Quesenbery defines usability using the following 5 dimensions:

- **Effective:** How completely and accurately the work or experience is completed or goals reached.
- **Efficient:** How quickly this work can be completed.
- **Engaging:** How well the interface draws the user into the interaction and how pleasant and satisfying it is to use.
- **Error tolerant:** How well the product prevents errors and can help the user recover from mistakes that do occur.
- **Easy to learn:** How well the product supports both the initial orientation and continued learning throughout the complete lifetime of use.

In this paper usability will be referred to an outcome. Where a UCD approach and UCD techniques will be used to ensure a systems usability.

Benefits of usability and UCD

There are many benefits in improving in using UCD and improving usability, they include:

- ### Gaining a competitive advantage

  When we improve the usability of our systems we make our products more attractive to users. Will a large number of users migrate to our systems whilst there are easier to use products available elsewhere? Possibly not. UCD has become big business with many large companies investing heavily in usability and in developing products that are easier to use than their competitors. If we are to compete, can we afford not to be engaging in UCD?

  Many FOSS systems have the rather large competitive advantage of being free. Is being free enough to attract or retain users if the system has poor usability? Will users prefer to surrender some of the freedoms FOSS offers buy pay for a more usable closed system?

- ### Increase customer satisfaction and loyalty

  IBM state that products that are easy to use lead to increased customer satisfaction, and satisfied customers return again and again. Ease of use also increases customer loyalty meaning they will be less likely to be looking seriously at your competitors for an easier option.

- ### Resolve internal design conflicts

  I'm sure everyone at some point in time has been involved in some heated discussions over whose design choices are better. UCD can take the ego and the opinions out of the equation. Through testing different designs with users we can gain a better understanding of which designs cause the least issues with users. We might find that some things work well in both designs while other parts don't. Maybe the best way forward is to discard what doesn't work and combine what does and test again.

- ### Save developers time

  Increasing a systems usability can decrease the number of support emails and posts to discussion boards. This allows developers to spend more time developing the system than troubleshooting and providing support for it.

  Developing designs and evaluating them can reveal features and functions which may not have previously been considered. This saves developers time as adding new features and functions into a system that's almost completed can take considerable time and heartache.

Organisation developing FOSS systems for profit will also realise the following benefits from performing UCD:

- ## Return on investment (ROI)

  ROI on usability is sometimes difficult to calculate. Despite this IBM state that as a rule of thumb every dollar invested in ease of use returns $10 to $100. They also warn that by skipping ease of use in the design phase organisations can "end up spending 80% of their service costs on unforeseen user requirements down the road".

  Nielsen (2003) urges project teams to spend 10% of their budget on usability. In research into web redevelopment they found that there was a 135% increase, on average, in usability. This increase lead to increases in sales and increased the number of visitors.

- ## Increasing revenues, decreasing costs

  Garrett (2003) points out that "in any commercial enterprise, Web sites exist for one of two reasons: to help the organization save money, or to help it make money. In both cases, the user experience can make the difference between a successful site and a failure." Bunnyfoot (2002) further elaborates by discussing how improving usability increases revenues and decreases costs:

  - *Increasing revenue:*
    - Decreased learning time
    - Decreased error rates
    - Increased efficiency
    - Increased user enjoyment and trust
  - *Decreasing costs:*
    - Speed up the design process
    - Avoid expensive modification costs
    - Decreased support costs
    - Decreased marketing costs
    - Decreased cost per sale

UCD techniques

There are many techniques that can be described as user-centred. These are just a few commonly used techniques.

Usability testing

Usability testing is one of the most commonly heard of UCD techniques. In usability testing designs are used by real users performing a number of tasks. The tasks are created to test features in the system and to make sure key user and organisational goals are met. By observing users we can see if there are any aspects of the design which are hampering ease-of-use and causing user dissatisfaction. From the issues found we can then start thinking about solutions to fix them.

Usability testing is an **iterative process** where we design, evaluate and then repeat. Through testing our designs and redesigns we can evolve the design making it more usable.

Nielsen (2000) recommends that 5 participants is a good number to have in a usability test. This number is recommended because through research they found:

- with 5 participants 85% of the usability issues were found;
- there are diminishing returns from testing with more users; and
- testing a small number of users will hopefully mean that more rounds of user testing can be performed.

With testing one user, whilst you can find some usability issues, there is a risk that you can walk away thinking that the issues faced by this user will be encountered by all, or at least a majority, of other users. This may or may not be the case. Testing a few users helps to better determine if these issues are representatives or just anomalies.

There may be clearly defined user groups that use your system. If you wish to make statements about the issues faced by particular audience groups it is recommended to test 3-4 users per group. As before just testing one user could mean you may jump to conclusions and make assumptions about the larger population that may or may not be founded.

Usability testing takes time and costs money. These could pose potential barriers to adoption and are discussed below in the section covering present open source usability efforts. Other challenges facing the adoption of UCD within the FOSS community are also discussed.

Paper prototyping

Paper prototyping is another form of usability testing. Instead of observing users using a live system, users interact with a paper based version. Paper prototyping allows you to test your designs early in the development process. Updates to paper designs can be made very quickly and cheaply.

In a paper prototype usability testing there are 3 key people:

- the **user**;
- the **facilitator** who administers the test and records any issues the user is having; and
- a colleague who plays the **computer**. They will be responsible for changing the screens, showing dialog boxes and drop-down menus depended on the users action.

In paper prototyping you have many individual pieces of paper representing the different screens and dialog boxes that make up the system. The more complex and larger the system is the more pieces of paper you'll have. Drop-down menus within your system are represented using sticky notes.

It is impossible to create a screen to account for every action the user may make. By making key screens generic and using squiggly lines the facilitator can ask the user to imagine that this screen contains the information appropriate to past information they have enter/manipulated.

In paper prototyping a pen is used in place of the mouse and keyboard. The user point at objects on the screen they wish to click and hand writes text into input fields.

Whilst there are certain user interaction modes that are difficult to implement in a paper based version paper prototyping can be an invaluable way of evaluating and then iterate your design to improve it's usability.

Card sorting (open/closed)

When developing a website or a menu/tabbed structure in a program we make choices about how we are going to group and label different options. Card sorting is a technique which allows us to collect information about how users think options should be group and labeled. It also allows us to test the labels we've created. Why is this important? If the options are grouped and labeled in a meaningful way, users will have less difficulty finding what they are looking for. Therefore improving the usability of our website/application.

In card sorting each different option is represented by an individual card. Each card has a title and may or may not have a very brief description about the option.

There are two types of card sorting:

- **Open card sorting**

  Users are given a pile of cards and asked to group them into meaningful groups. Once they have created their groups the facilitator then prompts them to create labels for all of these groups and often to prioritise the groups.

- **Closed card sorting**

  As with open card sorting users are given a pile of cards but also given a second pile of different coloured cards with group labels written on them. Users are then asked to sort the cards under these group labels. Closed card sorting allows us to test the groups and labels we've previously created. If users have difficulty understanding the labels and group the options differently to how we have this shows they may have difficulty locating these options in a developed version of our system.

Card sorting can be performed individually, in pairs, or in small groups. Nielsen (2004) recommends testing 15 users. By getting users to talk aloud during the sort rich qualitative data can be collected to compliment the quantitative data of how users grouped the cards.

Card sorting software is often used to collate and analyse the quantitative data. This software typically performs a cluster analysis and creates a dendogram which graphically shows the groups the users created and the strength of the association between the cards in the groups. The more individual users who grouped certain cards together the stronger the association is.

CardSword (http://cardsword.sourceforge.net/) is one open source card sorting program presently in development. Maurer (2004) has breakdown of other card sorting programs available (http://www.maadmob.net/donna/blog/archives/000583.html).

Card sorting is best performed early in the design process. While paper prototyping and usability testing allows us to gather feedback on created designs, card sorting allows us to gather information that can help inform the design process.

Contextual inquiry (AKA field studies)

Contextual inquiry is a technique that can be used very very early in the design process to gain a better understand of what problems users are facing. Beyer and Holtzblatt (1997) define contextual inquiry as “a field data-gathering technique that studies a few carefully selected individuals in depth to arrive at a fuller understanding of the work practice across all customers”.

The idea is that through using a master/apprentice or a partnership relationship model we can observe users in their own context and get a better understanding of the tools they use, the sequences of their actions, the way they organise information, and the kind of interactions they have with other people. Photos, video tapping, photocopies of documentation and handwritten notes are ways information is collected during an inquiry.

As you can imagine this technique can be quite time consuming and produce a lot of data. One technique for analysing the data is to produce an *affinity diagram* (AKA the KJ technique) where notes are created on individual post-it stickies and then groups and labeled by a group of people. From this exercise a number of models can be created to develop a concise picture of what has been observed.

What UCD isn't...

There are many ways that help with the development of user interfaces. Many of these techniques have the ability to improve the usability of your software but using them won't necessarily assure this. Whilst being rather obvious any technique that doesn't involve the user cannot be seen as user-centred. Without this feedback loop you are unable to accurately evaluate how easy to use your software is.

Each of the following techniques has it's merits and I am not saying don't use them. What needs to be remembered is that these techniques shouldn't replace UCD techniques in your projects. They can complement but should never replace them.

Let's just ask an expert

Getting advice from an experienced user interface designer can be very helpful. If they have been involved in performing a number of UCD techniques there are certain issues that they will have previously observed which may cause problems in your design. Also if interface design is their profession they will most likely be well versed in the appropriate literature which has been produced by other practitioners or through academic research.

The thing to be mindful of is that they are just suppling an opinion. Granted it may be an educated opinion, but at the end of the day it's an opinion none the less. Be careful of people just say “do it like this and it'll be more usable” without stating without testing it with users we won't truly know if this is the case.

Guidelines

Guidelines are recommendations on how desktop applications and websites/web applications look and feel. Rabourn (2004) defines guidelines as being "more specific than principles but less explicit than conventions, standards, or rules". Guidelines help capture user interface design knowledge/experience and pass it onto other designers.

There are many different guidelines available, some include:

- GNOME human interface guidelines
  http://developer.gnome.org/projects/gup/hig/
- KDE human interface guidelines
  http://usability.kde.org/hig/
- Apple human interface guidelines
  http://developer.apple.com/documentation/UserExperience/Conceptual/OSXHIGuidelines/
- Microsoft Windows User Experience
  http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwue/html/welcome.asp
- Web guidelines:
  - W3C Web Content Accessibility Guidelines 1.0
    http://www.w3.org/TR/WCAG10/full-checklist.html
  - Guidelines for Visualizing Links
    http://www.useit.com/alertbox/20040510.html
  - Top Ten Guidelines for Homepage Usability
    http://www.useit.com/alertbox/20020512.html
  - Error Message Guidelines
    http://www.useit.com/alertbox/20010624.html

Guidelines can help ensure consistence amongst different desktop applications. This helps users transfer knowledge from one application to another making them easier to use. But as Microsoft (2000) points out "consistency in itself doesn't ensure usability".

There is always the question of what guidelines should you follow? Some guidelines are created from user research but many are not. The same issues as discussed above can arise if a guideline is just based on somebodies opinion.

Applying guidelines can be quite challenging and can often lead to their misapplication. Even if guidelines are correctly followed the resulting system may not necessarily be usable.

Heuristic evaluation

In a heuristic evaluation a number of evaluators evaluate a particular design against a set of heuristics (eg. Nielsen 10 usability heuristics http://www.useit.com/papers/heuristic/heuristic_list.html). Usability problems found are aggregated and rated by the evaluators for their severity.

As with usability guidelines there are similar issues with heuristics and heuristic evaluation. They include:

- Which set of heuristics do you use?
- How were the heuristics created? Do the stem from user research or just opinion?
- Even if heuristics are followed there is the possibility that the system created will not be usable.

As heuristics are rules of thumb they are can be helpful when designing systems however they should not be considered a substitute for evaluating with users.

Personas

One trap that we can fall into is designing systems just for ourselves. This might be fine if the system is just for our personal use or for other people similar to us. The issue is when these systems are created for users who may come from very different backgrounds to us. Something we may think is obvious may be totally foreign to another group of users. Cooper (1999) discusses this problem and presents personas as a way of overcoming it.

Systems we develop are often created for different user from distinct user groups. Personas basically give a picture of what a stereo-typical user from one of these groups may look like. The persona may include a stock photo, a false name, some demographic information (eg. age, sex, income), their motivations, common things, websites they like to visit, etc.

There is a wide range of different information which can be included and formats the personas can take.

Personas are ideally created from information collected using user-centred techniques (field studies, surveys, interviews, etc.) and other market research techniques. As you can imagine persona creation is quite subjective. If not created from information collected from users they are just our best guess as to what we think our users are like. If they are just a best guess is the character in the persona really representative of a particular user group?

Once created personas can be distributed in poster format to those who are involved with designing and developing systems. The idea is that people will put the poster up in their workspace and gaze at it whilst developing the system. Instead of developing the system for ourselves we will gain empathy for our users and design for them in mind.

Personas can be a useful tool in communicating information about the users we are developing for. Sorry to sound like a broken record, but if real users aren't used to evaluation the system we've designed we are unable to determine how usable it actually is. As with the other techniques described above personas can complement the design process but should not be seen as a replacement for other user-centred techniques.

Open source usability efforts

Usability and UCD is growing in the FOSS arena. There are a number of initiatives and groups that are dedicated to promoting and performing UCD. A few include:

- OpenUsability
  http://openusability.org/
- BetterDesktop
  http://www.betterdesktop.org/
- GNOME Usability
  http://developer.gnome.org/projects/gup/
- KDE Usability Project
  http://usability.kde.org/

The following is a brief discussion of some of the challenges facing the adoption and use of UCD.

Lack of awareness as to what UCD and usability is

Whilst usability and UCD has been around for many decades it has really been the last decade that they have become more prevalent in software development. In the FOSS community there is a growing number of people who have heard about usability but there are also many who haven't. There are also those with a limited understanding of what it is.

People with a limited understanding of what usability is, may do certain things in the name of usability that don't include users which may or may not actually improve the systems usability. One of the key goals of this paper and presentation is to give people a better understanding of what usability and UCD is and to dispel some of the techniques which aren't particularly user-centred.

Part of the territory of being a usability professional is evangelising usability and UCD. This paper is also a call to arms to my fellow usability practitioners asking them to consider getting involved in the FOSS community. Consider donating some time to help mentor or perform UCD.

It takes time to perform

There are many different UCD techniques which range in the amount of time required to perform. Despite this it does takes time to organise, perform, and report findings and recommendations. With tight schedules and peoples desire to ship UCD can sometimes cut out of the time. As discussed earlier developing, testing and iterating paper based designs can be done very quickly and cheaply. Investing time in UCD can save development time spent ironing out issues users are having, or are likely to have with the interface.

Costs of performing usability (eg. compensating participants for their time)

Whilst certain usability techniques can be performed quite cheaply there is a cost involved in performing UCD. As previously discussed UCD can help:

- attract and retain more users to your system through greater user satisfaction and competitive advantage,
- solve design disputes,
- reduce the number of support calls and emails which frees up developers to do more of what they enjoy doing, and
- generate greater profits for organisation selling FOSS systems.

If you are committed to creating a quality system, to be used by a growing base of users with varying skill levels, please consider investing in UCD. The following is a break down of some of the typical costs.

In usability testing participants are often given a gift for their time. This is to thank them for giving up some of their personal time, especially if they have had to take time off work to participate. Some factors which influence the amount given include; the duration of the test, the amount of time taken to get to the test, and the professional status of the participant. Highly paid executives or consultants may require a greater incentive than say a student. Gifts can also help minimise the number of no-shows (scheduled participants who don't show up for the test).

To minimise this cost non-monetary gifts could be given. For example giving merchandise from that system (eg. branded t-shirts, toy mascots, etc.) or recognising the participants in the systems credits or on the systems website.

Testing materials like paper for consent forms, pre/post-test questionnaires, and paper prototypes is another typical cost. You may also need to pay a usability professionals for their time unless their services are offered pro bono through some of the above open source usability efforts.

Fitting UCD into FOSS development processes

Fitting UCD into the FOSS development process presents a great challenge. One that will require some close collaboration and clever thinking on behalf of usability professionals and members of the open source community. The question is how do you best integrate design and evaluation techniques, which are traditionally designed to precede development, into a development process that is characterised by its early, rapid, and distributed development and evolving design?

Work presently being done to bring usability and UCD to agile development processes (eg. eXtreme Programming) are applicable to the FOSS development process. Usability on these types of processes is being called 'agile usability'.

For further information on agile usability see Amblers' (2006) *Agile Usability: User Experience Activities on Agile Development Projects* article (http://www.agilemodeling.com/essays/agileUsability.htm) or join the Agile Usability Yahoo! group (http://tech.groups.yahoo.com/group/agile-usability/).

Further resources

In this paper and in the following bibliography there are many references for UCD and usability. If you are interesting in learning more about UCD and usability the following resources are recommended:

- Garrett, J. (2001), *The Elements of User Experience: User-Centered Design*, New Riders Press, New York.
- Snyder, C. (2003), *Paper prototyping : the fast and easy way to design and refine user interfaces*, Morgan Kaufmann, San Francisco.
- Kuniavsky, M. (2003), *Observing the user experience : a practitioner’s guide*, Morgan Kaufmann, San Francisco.
- Stone, D., Jarrett, C., Woodroffe, M., Minocha, S. (2005), *User Interface Design and Evaluation*, Morgan Kaufmann, San Francisco.
- Dey Alexander's user experience design resources http://www.deyalexander.com.au/resources/uxd/
- Jakob Nielsen's Alertbox web usability column http://www.useit.com/alertbox/
- Computer-Human Interaction Special Interest Group (CHISIG) http://chisig.org/

Bibliography

- Alexander, D. (2003), *Introduction to usability and user-centred design*, viewed 20 Sep. 2006, <http://its.monash.edu/staff/web/slideshows/usability-ucd/>.
- Ambler, S. (2006), *Agile Usability: User Experience Activities on Agile Development Projects*, viewed 13 Nov. 2006, <http://www.agilemodeling.com/essays/agileUsability.htm>.
- Beyer, H., Holzblatt, K. (1997), *Contextual Design: A Customer-Centered Approach to Systems Designs*, Morgan Kaufmann, San Francisco.
- Bias, R., Mayhew, D. (1994), *Cost-justifying usability*, Academic Press, London.
- Bunnyfoot (2002), The Business Case For Usability, viewed 20 Sep. 2006, <http://www.bunnyfoot.com/freestuff/articles/usability/usabilitybusinesscase.html>.
- Cooper, A. (1999), *The Inmates are running the asylum: why high-tech products drive us crazy and how to restore the sanity*, Sams, Indiana.
- Courage, C., Baxter, K. (2005), “Card sorting”, *Understanding your users: a practical guide to user requirements*, Morgan Kaufmann, San Francisco, pages 414 – 447.
- Dumas, J., Redish, J. (1999), *A Practical Guide to Usability Testing*, Intellect, Exeter.
- IBM, *Cost justifying ease of use*, viewed 20 Sep. 2006, <http://www-3.ibm.com/ibm/easy/eou_ext.nsf/Publish/23>.
- Garrett, J. (2003), *All Those Opposed: Making the case for user experience in a budget-conscious climate*, viewed 20 Sep. 2006, <http://www.ddj.com/dept/architect/184411634?pgno=1>.
- Kuniavsky, M. (2003), *Observing the user experience : a practitioner’s guide*, Morgan Kaufmann, San Francisco.
- Maurer, D. (2004), *Card sorting tools - a short summary*, viewed 17 Sep. 2006, <http://www.maadmob.net/donna/blog/archives/000583.html>.
- Memmel, T. (2006), *Agile Usability Engineering*, viewed 13 Nov. 2006, <http://www.interaction-design.org/encyclopedia/agile_usability_engineering.html>.
- Microsoft Corporation (2000), *UI Guidelines vs. Usability Testing*, viewed 13 Aug. 2006, <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwui/html/uiguide.asp>.
- Nielsen, J. (2004), *Card Sorting: How Many Users to Test*, viewed 16 Sep. 2006, <http://www.useit.com/alertbox/20040719.html>.
- Nielsen, J., *How to Conduct a Heuristic Evaluation*, viewed 13 Aug. 2006, <http://www.useit.com/papers/heuristic/heuristic_evaluation.html>.
- Nielsen, J. (2003), *Return on Investment for Usability*, viewed 20 Sep. 2006, <http://www.useit.com/alertbox/20030107.html>.
- Nielsen, J., *Severity Ratings for Usability Problems*, viewed 13 Aug. 2006, <http://www.useit.com/papers/heuristic/severityrating.html>.
- Nielsen, J., *Ten Usability Heuristics*, viewed 13 Aug. 2006, <http://www.useit.com/papers/heuristic/heuristic_list.html>.
- Nielsen, J. (2000), *Why You Only Need to Test With 5 Users*, viewed 20 Sep. 2006, <http://www.useit.com/alertbox/20000319.html>.

- Quesenbery, W., *Using the 5Es to Understand Users*, viewed 21 Sep. 2006, <http://www.wqusability.com/articles/getting-started.html>.
- Rabourn, T. (2004), *Making guidelines part of the team*, viewed 13 Aug. 2006, <http://www.boxesandarrows.com/view/making_guidelines_part_of_the_team>.
- Snyder, C. (2003), *Paper prototyping : the fast and easy way to design and refine user interfaces*, Morgan Kaufmann, San Francisco.
- Spool, J. (2002), *Evolution Trumps Usability Guidelines*, viewed 13 Aug. 2006, <http://www.uie.com/articles/evolution_trumps_usability/>.
- Spool, J. (2004), *The KJ-Technique: A Group Process for Establishing Priorities*, viewed 20 Sep. 2006, <http://www.uie.com/articles/kj_technique/>.
- Stone, D., Jarrett, C., Woodroffe, M., Minocha, S. (2005), *User Interface Design and Evaluation*, Morgan Kaufmann, San Francisco.