***The use of Heuristic Evaluation and Cognitive Walkthrough in User Interface and User Experience Testing***

**Leila Benhariz**

The usability testing methods of Heuristic Evaluations and Cognitive Walkthroughs are increasingly used in testing how a user may interact with a software application. They base their evaluations on how a human interacts with the user interface of an application and the resulting user experience. They check that the software accomplishes the intended use by testing the application with different scenarios of how a user may interact with it. This essay will discuss in detail the use of Heuristic Evaluation and Cognitive Walkthroughs in User interface and User Experience testing; how they are conducted; the advantages and the disadvantages; and I will also provide an evaluation of these two techniques in terms of their use and effectiveness.

Heuristic evaluation tests the design of the product and ensures that different usability evaluation techniques are developed and implemented into the design of the application (Tan, 2009). It is a method for defining usability problems throughout the user interface so that as the software is developed, they can be identified and fixed as part of an iterative design process (a process involving prototyping, testing, analysing and improving a software product repeatedly as a cycle until completion). In general, a number of usability testing experts will review the software and compare it against a list of pre-defined principles to ensure that the user experience and user interface is as desired. It would be difficult for a single individual to do the heuristic evaluation themselves, but if a number of people do it, it is far more likely that issues in the product will be found, improving the effectiveness of the evaluation (Nielsen, 1994).

The pre-defined principles are known as the heuristics. There are a long list of heuristics that can be followed and used. While others do exist and may be more modern and concise, one of the most widely used set of heuristics for user interface design comes from Jakob Nielsen and Rolf Molich. They are as follows:

1. Visibility of system status

*“The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.”*

When a user performs some action, they should know whether the request is being processed by being given some sort of system status, such as a loading bar or a progress update. For example, on a MacOS machine, when a user interacts with an item within the interface, they are presented with a pin wheel animation representing that the system is loading a response. Otherwise, a user can be left confused and dissatisfied if they cannot tell if their desired intention is being returned.

2. Match between system and the real world

*“The system should speak the users’ language, with words, phrases, and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.”*

The user should be able to easily understand the system, which can be achieved by making the user interface readable and clear. Make the designs and interactions more familiar to the real world, with less of a coding and more of a friendly feel to it which is understandable in natural language.

3. User Control and Freedom

*“Users often choose system functions by mistake and will need a clearly marked “emergency exit” to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.”*

A user needs to have control over the current state of the product and be given the appropriate options when the wrong input is given on their behalf. If they want to return to a previous state, allow them the function to do so.

4. Consistency and Standards:

*“Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.”*

Many different software’s use the same style which users become familiar with across different products. Follow this template so that users can easily navigate the interface. For example, most web pages have a log in option on the top right of the page. If you put it somewhere else like the bottom, the user could get confused.

5. Error Prevention:

*“Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.”*

If a user makes a mistake, make sure they are made aware of this issue and provide them with a prompt guiding them towards correcting it, rather than not helping them which would be baffling.

6. Recognition rather than recall:

*“Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.”*

Showing users things that they can recognise improves usability over needing to recall items from a previous use of the product as the extra content helps users retrieve information from memory. (Budiu, 2014)

7. Flexibility and efficiency of use:

*“Accelerators—unseen by the novice user—may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.”*

Providing expert users with options with options to filter and sort data in the product makes it easier for them to find what they are looking for, while novice users can still use the product without issue. For example, a retail site could offer different filters that users who are familiar with the system could use to find the item they are looking for, while any other user could still find it too, just at a slower pace.

8. Aesthetic and minimalist design:

*“Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.”*

There is no need to add extra elements that do not provide any meaningful information for the user. Only have relevant dialogues within the interface.

9. Help users recognise, diagnose, and recover from errors:

*“Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.”*

Provide the user with acknowledgment that something went wrong within the system and give them suggestions to rectify the problem, possibly with links for support or returning to a previous state of the product.

10. Help and Documentation:

*“Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.”*

If it is necessary for the user, provide them with helpful information but only if it is required. There is no need to give this information within the interface of the product if it is not sought after.

Once the list of heuristics have been selected, a set of evaluators are chosen to test these elements. They could include specialists, developers, the targeted user themselves, and it is not necessary to have a relatively large group to carry this out. A method for the evaluation is then set out. It is important as the evaluation process is carried out, to possibly develop a new set of heuristics more relevant to the product rather than following the likes of Nielsen’s list. Once the heuristics evaluation is completed, the problems are recorded and another evaluation phase could be set out to apply a new set of heuristics.

Heuristic Evaluations have numerous advantages and disadvantages. It can provide quick, valuable, and cost-effective feedback to the product developers. It allows feedback to be obtained early on in the design process. The best corrective measures can be derived from assigning the correct heuristic. Heuristic evaluations can also be used alongside other usability testing methods. On the other hand, to apply the heuristics effectively, it requires knowledge and experience, with trained usability specialists difficult to find and they can be costly. You need to use multiple experts and combine all of their results. The analysis in the end can identify more minor issues and fewer major issues. (Usabilty.gov, 2020)

On the other hand, cognitive walkthroughs are also carried out. Cognitive walkthroughs were initially used in the early nineties and have been popular among user interface and user experience testers since. In comparison to heuristic evaluations, cognitive walkthroughs are a task-specific approach to usability. Cognitive walkthroughs are a user-interface design technique that helps designers’ model how a type of user will understand an interface and how they will interact with it to achieve their desired outcome (Farrell and Moffat, 2014). The tester gets into the mindset of a user and tests the interface with how the user will interpret it in mind. The idea is that users prefer to learn how to navigate a system by actually using it to carry out certain tasks, rather than having to read instructions. Cognitive walkthroughs ensure this is possible for the user.

To conduct a cognitive walkthrough, you begin by defining a set of tasks or scenarios that an identified user would be likely to carry out and how you would expect the product to respond. Then the developer would take a step-by-step approach to examining the product while trying to imagine the user’s perspective of it. The goals and the process of the walkthrough should be considered, along with a possible set of rules to be followed. A list of scenarios and questions could be put together to do this, allowing the designer to compile a report of potential issues the user could have while interacting with the user interface. While walking through each scenario, typically four questions are posed:

1. Will the user try and achieve the right outcome?

The interface is tested with this question in mind to see if it makes assumptions about the user like how knowledgeable they are about how the system works. If you keep previously discussed heuristics in mind, it helps to recognise whether or not the interface is easily understandable for the user.

2. Will the user notice that the correct action is available to them?

This ensures that all of the necessary controls and options are visibly available to the user. If there is too much information on the page, the user can get confused, hindering the user experience. If there is not enough, the user could get lost in the interface being unsure where to go to achieve their intended use of the product.

3. Will the user associate the correct action with the outcome they expect to achieve?

This question asks if the product is clear for the user to understand. Does it use natural language and familiar icons that the user can navigate through to achieve their desired outcome? Or is it confusing and foreign to them.

4. If the correct action is performed, will the user see that progress is being made towards their intended outcome?

The developer can check that the user will know the status of what is currently happening when they try to interact with the product. It checks if feedback is being given to the user so that they know there is progress within the system.

By answering these questions, usability issues can be identified and compiled. Once this is done, the results of what was learned through the walkthrough can be considered and them the software can be adapted to address these problems. It is often best if the cognitive walkthrough is carried out by someone who is not involved in the design of the product as they would be less of a designer and more of a user, being able to identify more usability issues and offer a different perspective.

Cognitive walkthroughs are essential as they are good at breaking down a system into different classes and scenarios to identify usability issues. The main advantage is that it shows the difficulty of learning how to use an interface on the users’ behalf without reading any documentation. They help identify problems with the user interface earlier in the design process quickly and efficiently while being low costing. Some problems can occur, however, in cognitive walkthrough testing. If the developer is conducting the analysis, they could already know how to achieve a certain task and show how to do it within the walkthrough, whereas in reality it could actually be difficult for the user to do themselves. If this occurs, the problem might not get fixed, which negatively affects the user experience. Unlike heuristic evaluations, which could potentially ask real world users to carry out the analysis, cognitive walkthroughs often do not test real users on the product, so their opinion will be missed.

Thus far, I have reviewed both heuristic evaluations and cognitive walkthroughs in the context of user experience and user interface testing and I have shown their importance in the development of a software product. Both analysis methods prioritise understanding how a user would perceive a software product and develops an understanding of how the software can be improved. They both ensure that the finished product ends up to the quality and usability standards the user expects to see. To summarise and recap, heuristic evaluation tests the design of the product and ensures that different usability evaluation techniques are developed and implemented into the design of the application. It asks a number of questions about the product known as the heuristics, which are the pre-defined principles to ensure that the user interface and user experience are positive and the needs of the user are met. Cognitive walkthroughs are a user-interface design technique that helps designers’ model how a type of user will understand an interface and how they will interact with it to achieve their desired outcome. They get into the mindset of the user and ensure that the user can achieve their desired outcome, without the need for explicit directions and instructions.

My personal evaluation of these two techniques are that a positive user experience would not be possible without implementing heuristic evaluations and cognitive walkthroughs. Having carried out a cognitive walkthrough before, it was clear to me that it is very difficult and inefficient for a designer to complete a product without considering the needs of the users in this way. A product could seem complete in the eyes of the developer yet be missing a number of features the user could be looking for or be difficult for the user to understand and use. Thus, to conclude, the use of Heuristic Evaluation and Cognitive Walkthrough in User Interface and User Experience Testing cannot be underestimated and should always be carried out throughout the development of a software product.

**References:**

* Assistant Affairs, n.d. *Heuristic Evaluations And Expert Reviews | Usability.Gov*. [online] Usability.gov. Available at: <https://www.usability.gov/how-to-and-tools/methods/heuristic-evaluation.html> [Accessed 19 May 2020].
* Bodiu, R., 2014. *Memory Recognition And Recall In User Interfaces*. [online] Nielsen Norman Group. Available at: <https://www.nngroup.com/articles/recognition-and-recall/>
* Farrell, D. and Moffat, D., 2014. Adapting Cognitive Walkthrough to Support Game Based Learning Design. *International Journal of Game-Based Learning*, 4(3).
* Nielsen, J, 1994. *Heuristic Evaluation: How-To: Article By Jakob Nielsen*. [online] Nielsen Norman Group. Available at: <https://www.nngroup.com/articles/how-to-conduct-a-heuristic-evaluation/>
* Tan, W., Liu, D. and Bishu, R., 2009. Web evaluation: Heuristic evaluation vs. user testing. *International Journal of Industrial Ergonomics*, 39(4), pp.621-627.