Evaluation of Subject-Specific Heuristics for Initial Learning Environments: A Pilot Study

Fraser McKay

School of Computing
University of Kent
fm98@kent.ac.uk

Michael Kölling

School of Computing
University of Kent
mik@kent.ac.uk

Keywords: POP-I.B. design of environments; POP-II.A. novices; POP-V.B. interviews.

Abstract

Heuristic evaluation is a "discount" technique for finding usability problems in well-established domains. This paper presents thirteen suggested heuristics for initial learning environments (ILEs). To investigate the usefulness of these heuristics to other developers, we conducted a pilot study that compared two groups of evaluators: one using an older, generalised set of heuristics from the literature, and one using our domain-specific heuristics. In this study, we compare not just the number of problems found, but the way in which the problem reports were expressed. There was a significant difference in the length of written comments when problems were found (those from the new set being longer). New-set reviews touch on more themes – many make suggestions about what would improve the problem; many comments refer to a suggested cause-and-effect relationship. As designers, we find this detail helpful in understanding problems. Quantitative data from this study is not large enough to support any robust conclusions about the relative thoroughness of the heuristics at this time, but we plan to use lessons learned from this study in a larger version shortly.

1. Introduction

There are many initial learning environments (ILEs), using different languages, available for educators to choose from. With the increase in the number of systems on the market, it is becoming increasingly important to be able to make informed decisions, backed by formal argument, about the relative quality of these tools. We advocate a significant increase in more formal or semi-formal evaluations of the quality of educational programming systems.

Heuristic evaluation is a method that is both useful and practical to make such an assessment of many aspects of educational programming systems. Some of the most frequently used sets of heuristics are application-area neutral. They specify goals and guidelines for software systems in general. However, these do not cover all of the problem areas that we are aware of in novice programming. In this case, application-area specific requirements can be included in the heuristics, and deeper insights might be gained. This can be done for a variety of application areas, and has been attempted for programming before (Pane & Myers, 1996; Sadowski & Kurniawan, 2011), though these heuristics have not then been evaluated, or validated, themselves.

In this paper, we propose a set of heuristics specific to ILEs. These heuristics should improve heuristic evaluations of such systems by combining a number of relevant aspects and criteria not formulated in any other set of heuristics. The proposed heuristics include aspects of general usability, as well as aspects (e.g. motivational and pedagogical effects) relevant to our specific application domain. To investigate their usefulness for other developers, we conducted a pilot study that compared two groups of evaluators: one using a popular, generalised set from the literature, and one using our domain-specific heuristics. In this study, we compared not just the number of problems found, but the way in which the problem reports were expressed. As designers, we find this detail helpful in understanding problems. Quantitative data from this study is not large enough to support

any robust conclusions about the relative effectiveness of the heuristics at this time, but we plan to use lessons learned from this study in a larger version soon.

1.1. Background

Heuristic evaluation was introduced as a "discount" method of analysing user interfaces without full user testing (Nielsen & Molich, 1990). When performing a heuristic evaluation, experts compare interfaces to sets of heuristics – general principles that describe aspects of an ideal system. Nielsen lists ten separate heuristics, formulated in the style of positive guidelines such as "The system should always keep users informed about what is going on, through appropriate feedback within reasonable time." (Nielsen, 2005b).

Heuristics can also be used in designing interfaces. In this case, designers pay conscious attention to the set of heuristics during the design process – taking Shneiderman's "Eight Golden Rules", for example (Shneiderman, 1997).

Nielsen's heuristics are intended to be generally applicable to a wide variety of software interfaces, and other authors have identified more specialised, domain-specific sets of heuristics. Pane & Myers (1996) defined a set of heuristics aimed at ILEs. These extend Nielsen's set with additional heuristics aimed at identifying issues with problems specific to this target domain.

2. Related work

2.1. Cognitive Dimensions and CD questionnaire

The research presented here, being concerned with programming, also conceptually overlaps with the Cognitive Dimensions (CD) framework (Green, 1989). The dimensions describe concepts relevant to cognition and HCI in programming notations, but – unlike heuristics – are not phrased as instructions. There are some attempts in the literature to use them as heuristics (Sadowski & Kurniawan, 2011). There are structured forms of cognitive dimensions evaluation (Blackwell & Green, 2000; Blackwell & Green, 2007), but they are, overall, less restrictive than some heuristic evaluation research (in the "classical" Nielsen sense, at least). Open-ended comment is more encouraged. The cognitive dimensions provide descriptions of concepts such as "viscosity" (resistance to code changes) and "secondary notations" (such as colour and spacing) - a vocabulary for researchers to use when discussing systems.

2.2. Testing/evaluating inspection methods

Heuristic and CD evaluations, like other "expert-based" methods, rely heavily on human evaluators' findings; in this case, driven by a set of heuristics. Therefore, it is critically important that the heuristics are valid in themselves. Validity is generally taken to mean "shown to be useful in uncovering actual usability problems" (Blandford & Green, 2008). Pane & Myers's heuristics are argued from the literature (of the time), but are not evaluated in practice. Nielsen & Molich's original heuristics were tested in user studies to refine them and to assess their validity (Nielsen & Molich, 1990). Hartson, Andre & Williges (2001) present a method of comparing evaluation methods, including three standard metrics (thoroughness, validity and reliability) based on other work. Sears (1997) formally defines the concepts of thoroughness and validity for evaluating inspection methods. Thoroughness is a measure of how many problem candidates, in a set of real "reference problems", a method finds. Validity is a measure of how many of the candidate problems found are "real" problems (as opposed to false positives).

3. The new heuristics

The need to define new domain-specific heuristics arose during an on-going effort to design a novel beginners' programming tool.

To aid in the design of the new system, we initially applied existing heuristics as design guidelines. We also conducted evaluations using the cognitive dimensions framework, in their role as discussion

tools, to study some existing ILEs. Pane & Myers's heuristics seemed a good candidate tool for our purposes, at first, since they are specifically designed for these kinds of system. However, a major drawback we encountered with these heuristics is their length. Using them in depth, they produced very long – and quite cumbersome, thus less useful – evaluations. In total, Pane & Myers have 29 heuristics, grouped under eight of Nielsen's headings. Differences between the heuristics were not always clear-cut; there is duplication across some categories, and using them to categorise some of the problems is distinctly difficult. Some problems seemed to fit two categories equally well, and others did not neatly fit any. Yet, when using Nielsen's original heuristics, in order to avoid the length of Pane & Myers', the evaluation missed some crucial areas that we consider important for early learners' programming systems. While shorter, these heuristics do not achieve the same domain specific precision.

This experience led to the definition of a new set of heuristics with a number of specific goals of improvement. The goals were:

(1) Manageable length: The number of heuristics should be limited; it should be much closer to Nielsen's 10 rules, or Shneiderman's eight, than Pane & Myers' 29.

(2) Domain-specific focus: The heuristics should cover aspects specific to novice programming.

(3) Avoidance of redundancy and overlap: As much as possible, problems should fit clearly into a single category.

(4) Clarity of categorisation: The categories should be clear to evaluators, cover all possible issues and be easily distinguished.

Blandford and Green (2008) suggest requirements that a new HCI research method should be useful ("it should reveal important things…"), usable, and used ("would ultimately be used by people other than its developers and their close friends").

After identifying target candidates for heuristics based on known problem instances, the authors evaluated the resulting sets by applying them to existing systems from the target domain. This process was repeated over several iterations to refine and improve the set.

The systems used for systematic evaluation of the heuristics were Scratch, Alice, BlueJ, Greenfoot and Visual Basic. Partial evaluations using the new heuristics have also been carried out for Lego Mindstorms NXT, StarLogo TNG, Kodu, and C#. The first group of systems are the ones that were used for testing each iteration of the heuristics; the second group are systems that we have evaluated using the heuristics. We have also used the heuristics to evaluate new designs, which we are working on as part of the wider project to design a new system.

3.1. New set

There are thirteen heuristics in our set. They are used to evaluate a complete ILE, including the environment aspects and the programming language aspects. The new heuristics are neither sub- nor superset of any previously existing set, and have been developed from scratch using well-established principles from the literature in this development domain.

(1) **Engagement**: The system should engage and motivate the intended audience of learners. It should stimulate learners' interest or sense of fun.

(2) **Non-threatening**: The system should not feel threatening in its appearance or behaviour. Users should feel safe in the knowledge that they can experiment without breaking the system, or losing data.

(3) **Minimal language redundancy**: The programming language should minimise redundancy in its language constructs and libraries.

(4) **Learner-appropriate abstractions**: The system should use abstractions that are at the appropriate level for the learner and task. Abstractions should be driven by pedagogy, not by the underlying machine.

(5) **Consistency**: The model, language and interface presentation should be consistent – internally, and with each other. Concepts used in the programming model should be represented in the system interface consistently.

(6) **Visibility**: The user should always be aware of system status and progress. It should be simple to navigate to parts of the system displaying other relevant data, such as other parts of a program under development.

(7) **Secondary notations**: The system should automatically provide secondary notations where this is helpful, and users should be allowed to add their own secondary notations where practical.

(8) **Simplicity/Clarity**: The presentation should maintain simplicity and clarity, avoiding visual distractions. This applies to the programming language and to other interface elements of the environment.

(9) **Human-centric syntax**: The program notation should use human-centric syntax. Syntactic elements should be easily readable, avoiding terminology obscure to the target audience.

(10) **Edit-order freedom**: The interface should allow the user freedom in the order they choose to work. Users should be able to leave tasks partially finished, and come back to them later.

(11) **Minimal viscosity**: The system should minimise viscosity in program entry and manipulation. Making common changes to program text should be as easy as possible.

(12) **Error-avoidance**: Preference should be given to preventing errors over reporting them. If the system can prevent, or work around, an error, it should.

(13) **Feedback**: The system should provide timely and constructive feedback. The feedback should indicate the source of a problem and offer solutions.

We have categorised the heuristics into four sets: those that affect the **System** as a whole, the **Mental Model**, the **User Interface**, and **Interaction** with the system. The System and Mental Model categories relate mostly to functionality, while the User Interface and Interaction categories are concerned with the presentation of the system. Figure 1 shows the relationship of these sets.

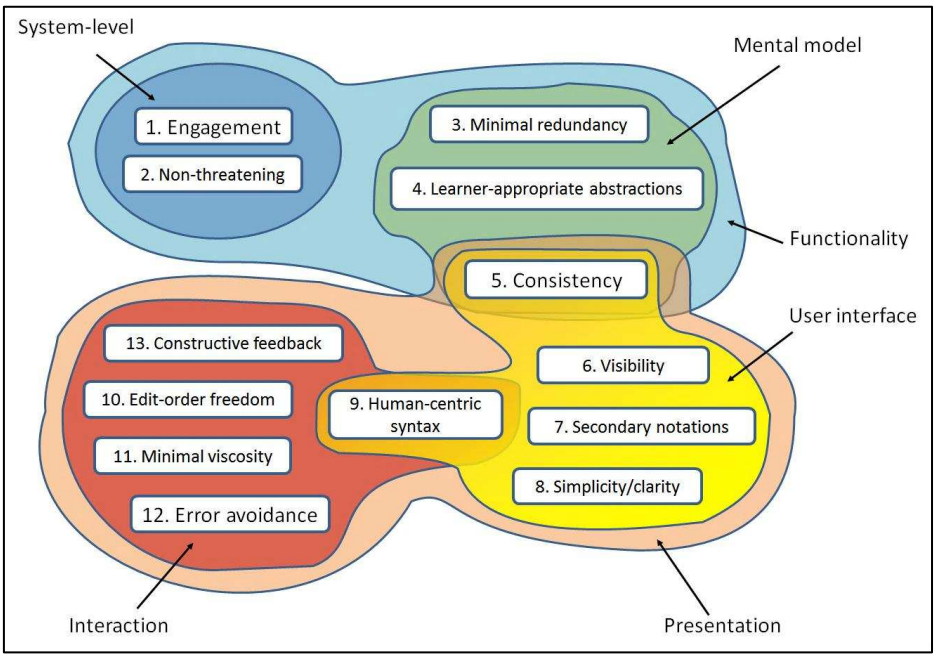

Figure 1. How the heuristics apply.

4. Method

Participants were asked to perform a heuristic evaluation of Scratch and Greenfoot, using a given set of heuristics. They were given tasks to carry out as part of their evaluation and record any usability problems they found. Each participant was assigned to use either the new heuristics, or Nielsen's heuristics (n=6, n=3, respectively). Participants were not told that the purpose of the study was to evaluate the heuristics themselves. They were not aware that other participants might use a different set of heuristics. All participants first completed the Scratch task. For various reasons, only five completed the Greenfoot task, so the Greenfoot task was excluded from this study. The participants are described in Table 1.

Participants were given a short "refresher" talk on the heuristic evaluation process, as part of their instructions. Between one and three participants at a time worked independently, in different parts of a large, quiet room. As well as a copy of the task, participants were given a sheet of paper listing the heuristics to use. The new heuristics were as presented above, and the existing set was taken from Nielsen (2005b). Both sets were formatted in the same plain style, and the title did not indicate anything special about there being multiple sets of heuristics. Participants were asked to record usability problems on paper pre-printed with columns for recording the problem, and which heuristic that problem breaks.

Twelve participants initially volunteered. However, due to three either withdrawing or not being available on the day, only nine participants actually participated in the study. The undergraduates were recruited through an email list following on from an HCI module. The postgraduates were recruited through a departmental email to research students. All of the participants were experienced programmers, and understood that some programming knowledge was required for the task(s). Though all knew of Scratch, only one had actually used it before – that reviewer had been taught using Scratch through secondary school. As seen later, the user discovered significantly more problems than others using the same heuristics.

We did not disclose until afterwards that the purpose of the study is to compare two sets of heuristics.

Reviewer	Profile	Heuristics	Interview
A	Undergraduate, had not used Scratch before	New	Y
B	Undergraduate, had not used Scratch before	New	Y
C	Postgraduate, had not used Scratch before	New	N
D	Postgraduate, had not used Scratch before	New	Y
E	Postgraduate, had not used Scratch before	New	N
F	Postgraduate, had not used Scratch before	New	N
G	Postgraduate, had not used Scratch before	Nielsen	Y
H	Postgraduate, had not used Scratch before	Nielsen	Y
I	Undergraduate, had long-term school experience with Scratch	Nielsen	Y

Table 1. Reviewer profiles.

At the end of the sessions, six participants who had time (three with each set) participated in short, informal individual interviews. The subject of the interview were the heuristics themselves, rather than the system under evaluation. The participants were asked open questions about ease or difficulty to understand and apply the heuristics; how they would describe the heuristics in their own words; and whether they thought any of the problems they found would have fitted with none of the heuristics, or more than one.

5. Results

5.1. Number of problems found

There was no statistically significant difference in the number of problems found by each group.

Sears's (1997) and Hartson et. al's (2001) validation methods rely on comparing "thoroughness" and "validity" per review(er). Using Equations 1 and 2, thoroughness and validity can be calculated for the Scratch problems (Table 3). However, the small size makes any real statistical analysis of t and v of little actual value in this case. In the nine results tested, there is little difference in thoroughness in the two groups. Measuring v does not produce anything notable with these results – none of the comments were clear "false positives". If the number of false positives is low (but non-zero), it would also have little or no reliable predictive value in this small study.

Group	N	Median	Avg Rank	Min	Max
New	6	0.177	5.167	0.065	0.290
Old	3	0.161	4.667	0.065	0.258

Table 2. U-test of thoroughness (p=0.794).

$$t = \frac{\#\ Real\ problems\ found}{\#\ Total\ problem\ set} \qquad v = \frac{\#\ Real\ problems\ found}{\#\ Suggested\ problems\ found}$$

Equation 1 and Equation 2.

Figure 2 is a diagrammatic representation of problems found per user. In the figure, every column represents a usability problem, and every row represents one reviewer. If that reviewer found the particular problem, the grid is marked with a black square. The first six evaluators used the new heuristics, and the last three used Nielsen's set (marked by a horizontal line). Similar diagrams are used in Nielsen's online material (Nielsen, 2005a). Problems to the left of the vertical centre-line were found by more than one evaluator. Problems on the right were only found by one person. Problems are numbered (horizontal, bottom).

Figure 3 summarises the problems found by each heurstics set. Any problem which was found at least once, for the given set of heuristics, is marked in black. Some problems were found with only one set, and not the other. A total of 9 problems were identified only with the new set (problems 6, 8, 10, 14, 15, 16, 17, 19, 21). Four problems were found only with the Nielsen set (problems 12, 13, 18, 20). Of problems which were identified by at least two evaluators, three were uniquely identified using the new set (6, 8, 10), and none with Nielsen's set.

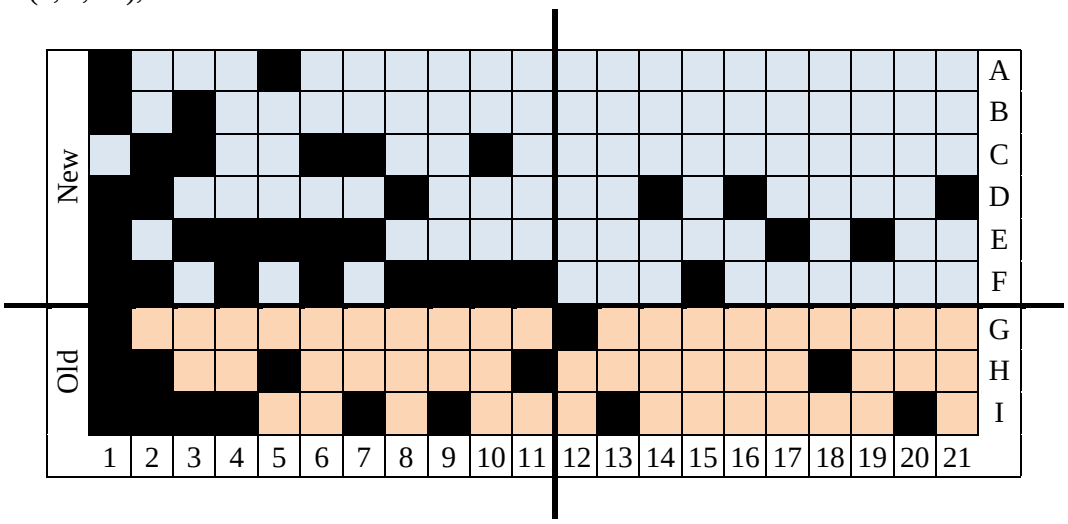

Figure 2. Problems identified per reviewer. Problems (horizontal) are numbered, *and participants'* (vertical) identifiers correspond to Table 1.

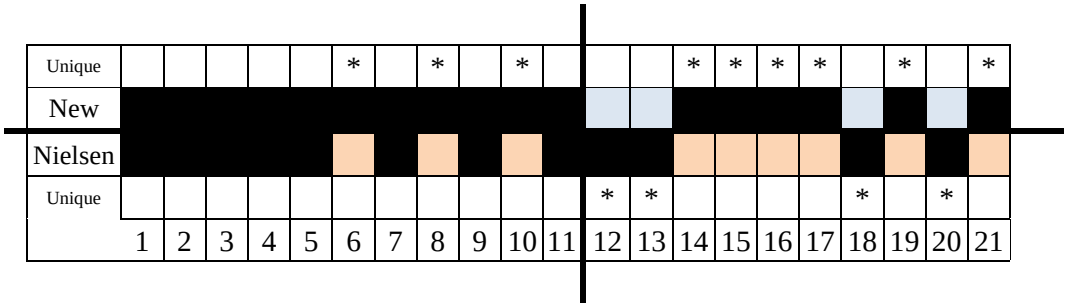

Figure 3. Problems identified per heuristics set. Problems found uniquely with one set are marked.

5.2. Comments made

The reviewers made 29 written comments, with the new heuristics; and 19 with Nielsen's set. The mean numbers of comments each were 6.0 and 6.18, respectively.

Though the number of comments per-reviewer was similar in most cases, the individual comments varied greatly in length (as well as giving additional qualitative information – see below). The values in Table 3 show the differences in numbers of characters. The new-set comments are generally longer.

Group	N	Mean	Min	Max
New	29	111.172	28.0	499.0
Nielsen	19	52.105	25.0	113.0

Table 3. Comment length by group (t-test p-value = 0.018)

Two typical comments, from different reviewers, are,

> "H8 simplicity. There is something strange with the way a whole block of blocks is moved, and one has to separate the first of these in."

> "H10, 8. I can not delete an [sic] statement by right clicking it if it's in between a sequence therefore I had to do 2 steps (separated them, deleted and put the rest back together)."

The longest comments with the new heuristics are paragraph-length. For instance,

> "H11. Dragging and dropping the blocks felt slow when doing a sequence of 10, doing the edit on both cars separately was annoying, and also replacing blocks was annoying, though I found a trick that helped (first add the new block beneath the old block, then drag it (and all thus all that follows it) to the right spot above the old block, then remove the old block which as it's the last item, doesn't uncouple the other blocks). Still, a right-click--delete option would have saved a lot of time."

In contrast, a longer comment from the Nielsen group reads,

> "H8. Some of the control colours are very similar, slowing down rate that you can guess where commands are stored."

(referring to Nielsen's heuristic 8 – aesthetic and minimalist design).

We are aware that having longer comments, per se, is not the aim of the new heuristics. However, in addition to being longer, the comments contained more detailed and descriptive feedback. Below, the comments are discussed in more detail.

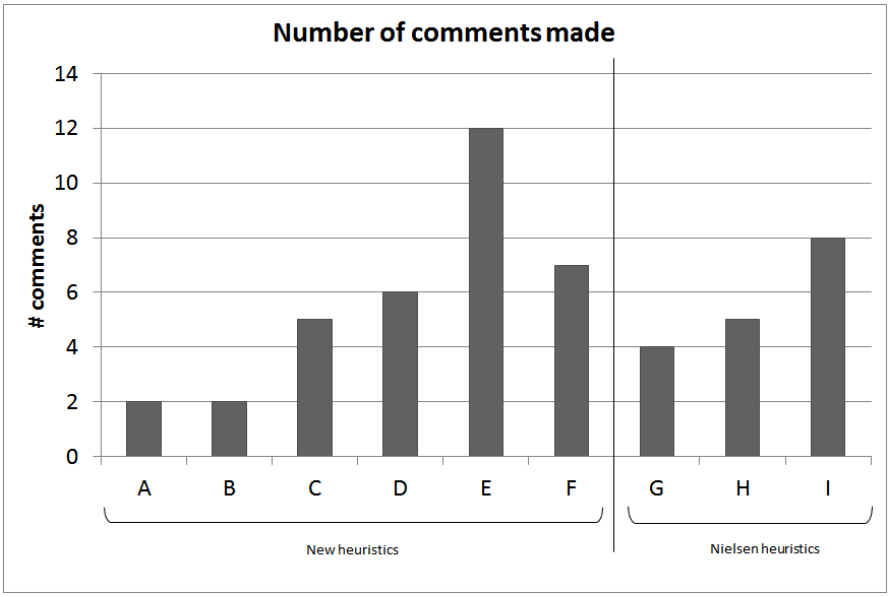

Figure 4. Written comment distribution.

5.3. Comment types

All the comments were coded: for the problems they described, and the nature of the comment. The coding for types of comment was open – there were no predetermined categories of response that we were determined to fit eventual results into.

Even though not explicitly instructed to do so, many of the evaluators included suggestions (sometimes quite detailed) in their problem reports. The following are examples of suggestions, all from the new heuristics (emphasis added):

> "Had to switch tabs a lot, **process would be more efficient if they were rearranged, or more visible at once**".
>
> [About there being a work-around to shuffling blocks] "**Still, a right-click delete option would have saved a lot of time**".
>
> "maybe have a way of integrating the left hand, e.g. by being able to switch between block-menus (which is done by clicking a button in the top-left square) with the key-board?".
>
> "[Would be] **Easier to make changes then duplicate rather than editing each one**".
>
> "**Maybe try, and balance the sprite pictures so there are a bit more girlish ones…**"
>
> "When the last change is compiled the button is still active (even though there is a state indicating). **However it would be better to block it**".

Reviewers made roughly the same number of comments using each set. However, there was a marked difference in suggestion frequency. There was only one suggestion made with the existing heuristics, out of 19 comments.

Some of the comments with the new set were phrased as cause-effect pairs. The problem was described as in other reports, but was also linked to an underlying issue or concept in the system. For example, the following three:

> "No types on the variables, **so** string/int confusion of 90/ninety could not be prevented".
>
> "[…] **as it's the last item**, doesn't uncouple other blocks".
>
> "The dragging blocks is very slow, [because] having to use the mouse a lot makes me use the right hand a lot more than the left."

5.4. Post-interviews

Qualitative data from the interviews refer to the nature of the heuristics themselves. Some of the comments queried the wording of particular heuristics (in both sets). For example:

- The wording of "intended audience" in the new set was ambiguous (it having been left open to reflect the range of systems it could apply to) (New set: 1);
- "Viscosity" (New set: 11) was a more difficult concept to grasp than, for example, error-proneness;
- "Aesthetic and minimalist design" (Nielsen: 8) did not quite convey, in their opinion, that "dialogues should not contain information which is irrelevant or rarely needed" (Nielsen, 2005b);
- They could not understand (or in one case, was sure there was none) the difference between "simplicity", in the sense of "visual simplicity" (New set: 8) and viscosity.

Although it was not the main purpose of the interview, a few of the participants pointed out an additional usability problem during interview. These were not solicited by the researcher, but typically expressed as something that the participant was unsure whether they should record, or say out loud in the interview. We decided it would be counterproductive to actively ignore a usability problem once it had been suggested; the researcher wrote on the interview notes that a problem was mentioned aloud. However, these are not included in the written comments being analysed here, and remain recorded as part of an actual review of the two systems, not the heuristics.

6. Discussion

6.1. Reflections on the study

Ultimately, this study is not large enough to definitively answer a question about problem coverage/discovery. The "evaluator effect" – the variability that necessitates using multiple reviewers – is well reported in the literature (Hertzum & Jacobsen, 2001). Within the nine participants in this study, we can see variation in how many problems different users found with the same heuristics. The small sample was then more open to skew. As mentioned, the Nielsen-group evaluator who had been taught to use Scratch at school found more problems than the others in their group. In a larger group, this might only have been a single outlier. Alternatively, it might have shown no relationship between specific prior experience and problem discovery. The evaluators from the other group who found a similar number of problems had not previously used Scratch. Quantitative data from this study is not large enough to support any robust conclusions about the relative thoroughness of the heuristics at this time.

However, this study does show that the new heuristics have found some actual, verifiable problems, at least as well as Nielsen's did, and possibly better. We know from the literature that different methods, like heuristics or user testing, will generally find different problems, and that a more complete picture of a system is obtained by inspecting it with multiple techniques (de Kock, van Biljon, & Pretorius, 2009; Hertzum & Jacobsen, 2001). In heuristic evaluations, the recommended number of evaluators varies (Cockton & Woolrych, 2001; Nielsen, 2005a).

Through qualitative analysis, we found extra kinds of helpful information embedded in the problem reports. This fits with the observation that problem reports made with the new set were significantly longer. In addition to comments describing a problem, there were many positively-phrased comments with both sets. These are not usability "problems", but it would be unproductive to discard them altogether. For the purposes of this paper, they are not included – in the graph of problem coverage, for example – but in a "real" evaluation they would certainly provide additional heuristics-based feedback on the system being tested. The "checkerboard" style summary used in a stricter test would obscure this entire category of information.

In those comments that did describe a problem, there were two "special" forms of comment observed: problems with suggested fixes; and problems where the reviewer linked two issues as having a cause-and-effect relationship. Examples of both were quoted in the results section of this paper.

The meaning of "viscosity" was an issue in several interviews. One participant, for instance, was quite confident that,

> "viscosity and simplicity are the same anyway."

The same participant described a problem related to viscosity (and correctly identified as such by other participants) using the word "simplicity", and categorised it under the heuristic relating to visual simplicity/clarity:

> "H8 simplicity. There is something strange with the way a whole block of blocks is moved, and one has to separate the first of these in."

Another interviewee understood that the word meant "thick", or "sticky", and understood what we meant by viscosity in this context, but did not think the choice of word "was quite right".

However, this was not universal – some clearly understood what viscosity meant. For example,

> "…first add the new block beneath the old block, then drag it (and all thus all that follows it) to the right spot above the old block, then remove the old block…"
>
> "Blocks "stick" together by default."

6.2. Heuristics as a set

The new heuristics may evolve in time, but they are adequate as a useful, and usable, starting point. We believe they are an improvement over Pane and Myers's (1996) draft set in the same domain. Some interview participants commented finding it difficult to categorise problems. Miscategorising is known to occur in heuristic evaluations, generally (Cockton & Woolrych, 2001).

It may be beneficial to investigate severity ratings – part of the "canonical" Heuristic Evaluation method by Nielsen, used sometimes but not always (Sim, 2011). We did not use them in this study because we felt they would simply complicate the comparison; we do not know if having to include a severity rating makes the evaluators more, or less, likely to report a problem, and whether this varies depending on the heuristic. It is also important that we see these heuristics as being useful during design, not just a traditional evaluation. In that case, it seems unnecessary for a designer to numerically rate problems that they have found themselves.

7. Conclusion and future work

Usability heuristics are used in the evaluation of a range of systems. This paper compares the characteristics of problem reports made using either new, domain-specific, heuristics, or an established set of generalised heuristics from the literature in the evaluation of an ILE. Within this pilot study, we found a significant difference in the kinds of comments reviewers made: reviews made with the new set typically included longer, more detailed, comments than those with the general set. Through interviews, we gathered the reviewers' opinions about the structure of the new heuristics.

As expected with a small group, the quantitative data is not definitive. The structure of this study proved that problems can be found, in these systems, with these sets of heuristics, and by these kinds of evaluators. An extended version of the experiment is planned later in the current academic year, using larger groups of evaluators.

8. References

Blackwell, A. F., & Green, T. R. G. (2000). A cognitive dimensions questionnaire optimised for users. Paper presented at the Proceedings of the Twelfth Annual Meeting of the Psychology of Programming Interest Group, 137-152.

Blackwell, A. F., & Green, T. R. G. (2007). A cognitive dimensions questionnaire. Retrieved, 2011, from http://www.cl.cam.ac.uk/~afb21/CognitiveDimensions/CDquestionnaire.pdf

Blandford, A., & Green, T. R. G. (2008). Methodological development. In P. Cairns, & A. L. Cox (Eds.), Research methods for human-computer interaction (pp. 158-174) Cambridge University Press.

Cockton, G., & Woolrych, A. (2001). Understanding inspection methods: Lessons from an assessment of heuristic evaluation. In A. Blandford, J. Vanderdonckt & P. Gray (Eds.), People and computers XV (pp. 171-192) Springer.

de Kock, E., van Biljon, J., & Pretorius, M. (2009). Usability evaluation methods: Mind the gaps. Paper presented at the Proceedings of the 2009 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists, 122-131.

Green, T. R. G. (1989). Cognitive dimensions of notations. People and Computers V: Proceedings of the Fifth Conference of the British Computer Society Human-Computer Interaction Specialist Group, 443-460.

Hartson, H. R., Andre, T. S., & Williges, R. C. (2001). Criteria for evaluating usability evaluation methods. International Journal of Human-Computer Interaction, 13(4), 373-410.

Hertzum, M., & Jacobsen, N. E. (2001). The evaluator effect: A chilling fact about usability evaluation methods. International Journal of Human-Computer Interaction, 13(4), 421-443.

Nielsen, J. (2005a). How to conduct a heuristic evaluation. Retrieved September, 2012, from http://www.useit.com/papers/heuristic/heuristic_evaluation.html

Nielsen, J. (2005b). Ten usability heuristics. Retrieved 08/12/09, 29, from http://www.useit.com/papers/heuristic/heuristic_list.html

Nielsen, J., & Molich, R. (1990). Heuristic evaluation of user interfaces. Paper presented at the CHI '90 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Empowering People, 249-256.

Pane, J. F., & Myers, B. A. (1996). Usability issues in the design of novice programming systems. (No. 96-132). Pittsburgh, Pennsylvania: Carnegie Mellon University.

Sadowski, C., & Kurniawan, S. (2011). Heuristic evaluation of programming language features. (No. UCSC-SOE-11-06). Santa Cruz, CA: University of California at Santa Cruz.

Sears, A. (1997). Heuristic walkthroughs: Finding the problems without the noise. International Journal of Human-Computer Interaction, 9(3), 213-234.

Shneiderman, B. (1997). Designing the user interface: Strategies for effective human-computer interaction (3rd ed.). Boston, MA, USA: Addison-Wesley.

Sim, G. (2011), Evaluating heuristics. Interfaces, 89, 16-17.