Towards Specifications for Automatic Recognition Software: An Example of a User-Centred Design

Sylvain Fleury, Eric Jamet, Emilie Loup-Escande, Achraf Ghorbel, Aurélie Lemaitre, Eric Anquetil

Journal of Software Engineering and Applications, 2013, 6, 1-4
http://dx.doi.org/10.4236/jsea.2013.610A001 Published Online October 2013 (http://www.scirp.org/journal/jsea)

Towards Specifications for Automatic Recognition Software: An Example of a User-Centred Design

Sylvain Fleury[1*], Éric Jamet[1], Emilie Loup-Escande[1], Achraf Ghorbel[2], Aurélie Lemaître[2], Eric Anquetil[2]

[1]Centre de Recherche en Psychologie, Cognition et Communication (CRPCC), University of Rennes 2, Rennes, France; [2]Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA), INSA Campus de Beaulieu, Rennes, France.
Email: [*]sylvain.fleury@uhb.fr

Received August 8th, 2013; revised August 30th, 2013; accepted September 5th, 2013

ABSTRACT

This article describes a user-centred method used to design innovative pattern recognition software for technical paper documents. This kind of software can make some errors of interpretation. It will therefore be important that human operators are able to identify and correct these mistakes. The identification of errors is a difficult task because operators need to establish co-reference between the initial document and it interpretation. Moreover, users must be able to checks the interpretation without forgetting any area. This task requires the interface is easy to use. The experiments showed that the sequential display of interpretation is the most effective and that the interruptions by user reduce task duration. Moreover, queries by the system may improve error detection. This paper summarizes the main results of the research conducted in the context of this design for enhance the interface, and describes the specifications to which it gave rise.

Keywords: User-Centred Design; Interpretation of Document; Error Detection

1. Introduction

There are currently several fields of application for pattern recognition software. For example, it can be used to recognize logic circuit diagrams, engineering drawings, maps, musical scores, architectural drawings and logos [1]. The automatic recognition of technical paper documents produces digital interpretations that are directly compatible with dedicated software. In this article, we describe a user-centred method we used to design innovative software that is capable of automatically interpreting hand-drawn architectural floor plans[1]. With this kind of interpretation software, there is always a risk of making mistakes. For instance, preliminary tests of our software revealed an error rate of 9% for simple plans [2]. It is therefore important for users to be able to identify these mistakes.

2. User-Centred Design

Two complementary approaches have been adopted in software design: a technocentric approach and an anthropocentric approach. The aim of the former is to design and optimize innovative software by testing its technological possibilities and resolving technical glitches. The aim of the latter is to design software that is adapted and adaptable to its end-users [3]. Ergonomics takes the anthropocentric approach, relying on human-centred design. The ISO 9241-210 [4] standard identifies six principles that characterize human-centred design: 1) the design is based upon an explicit understanding of users, tasks and environments; 2) users are involved throughout design and development; 3) the design is driven and refined by user-centred evaluation; 4) the process is iterative; 5) the design addresses the whole user experience; 6) the design team includes multidisciplinary skills and perspectives. The application of these principles gives rise to four main tasks. The first task consists in understanding and specifying the context of use. The second in specifying user needs and the other stakeholders' requirements, the third task in producing design solutions (e.g. scenario, mock-up, prototype), and the fourth task in assessing the solutions at each stage in the project, from the early concept design to long-term usage, in order to make the right design choices. There are two

[*]Corresponding author.
[1]This project, named "Mobisketch", is funded by the French National Research Agency (ANR). Its reference is 09-CORD-015 (http://mobisketch.irisa.fr).

main user-centred methods that can be used to carry out this fourth task, namely user-based testing and expert evaluation based on the usability literature. In our project, we opted for a combination of user-based testing and usability heuristics. These evaluation methods are complementary, as experimental user testing yields information about user behavior in a specific interaction with a designed solution, while the usability literature provides general knowledge about the human characteristics that need to be integrated in an interface. Together, these two evaluation methods would help us come up with precise specifications for designing software capable of automatically recognizing architectural plans. In the following sections, we summarize the main results of our research and describe the specifications to which it gave rise.

In a preliminary test, we asked 40 volunteers to check for errors of interpretation made by our initial prototype. The two pictures that had to be compared (the hand-drawn plan and its interpretation) were displayed side by side. In this seemingly simple task, only 33% of the volunteers succeeded in identifying all the errors. The specifications described below came out of a series of studies conducted during the user-centred design phase. They apply to the design of all automatic recognition or beautification software.

3. How Can We Facilitate Plan Comparisons?

Thirty-six participants were divided into two groups. They were told to circle the errors in three interpretations of three plans. For the first group, the plans and their interpretations were displayed side by side, whereas the second group worked on interpretations that were superimposed on the plans (see **Figure 1**).

Results failed to reveal any significant difference between the two groups in the accuracy of pinpointing errors. However, it took the participants who had to compare two separate images significantly longer to complete the task than those whose images were superimposed. This result is consistent with the principle of spatial contiguity, according to which distant visual information sources hinder learning [5]. Although our task involved error searching, rather than learning, this principle can still be used to explain our results. When the plan was separated from its interpretation, the participants had to store information from the plan, such as the location of a door, in working memory and find the equivalent location in the interpretation. This sort of visual searching is costly in cognitive resources and a waste of time for users. Superimposing the interpretation produced by the software on top of the original source reduces the visual searching and enables users to save time.

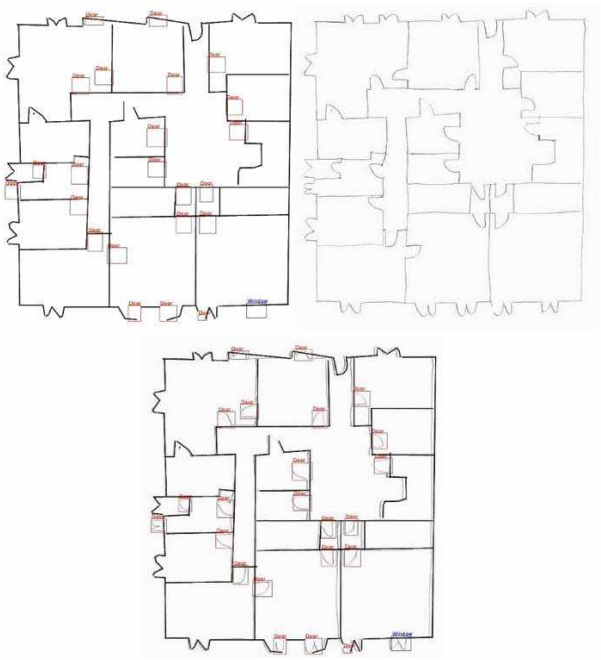

Figure 1. A plan and its interpretation displayed side by side (top) or superimposed (bottom).

4. How Can We Facilitate Error Detection?

A third group of 18 volunteers took part in a similar study, in which the interpretation gradually appeared on the screen, as and when each feature was recognized by the software. Results showed that the realtime display of the interpretation significantly improved the percentage of participants who spotted all the errors. The sudden appearance of an item on the screen triggers attentional capture [6]. In all probability, the sequential nature of the display meant that all the participants checked all the areas of the plan in turn, whereas had the interpretation appeared all of a sudden, they might have forgotten to check some of them. A replication of this study, supplemented with eye movement recordings, corroborated this interpretation of the results.

To assess the need for visual cueing, we conducted interviews with 18 participants after they had used the prototype. Many of them complained that the symbols did not appear on the screen in a logical order. For example, the software might interpret three symbols that were close together, then one that was located on the other side of the plan. Depending on the software's technical characteristics, it is sometimes not possible to modify the order of symbol recognition. When there is no obvious logic to this order, the software can implement a pre-guiding function, whereby the feature it is about to process is highlighted with a colour halo (see **Figure 2**). This attentional focus tells the user which symbol will be interpreted next.

This addition is consistent with the heuristic criterion

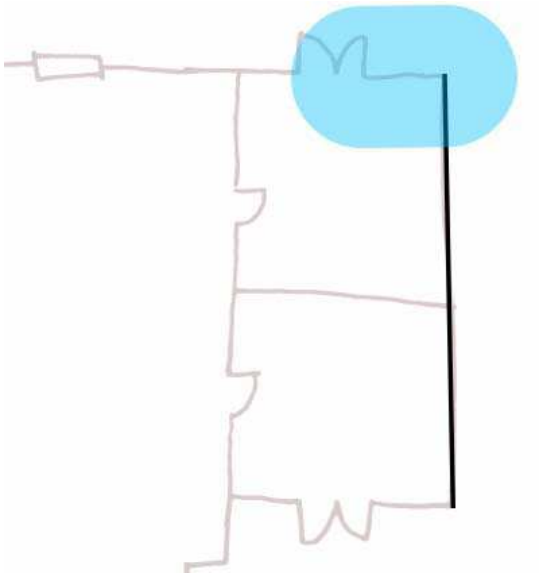

Figure 2. Visual cue indicating which symbol will be processed next.

of Molich and Nielsen [7], known as the "visibility of system status", which consists in showing the user what the software is doing. We also applied the "match between system and real world" criterion, by using intuitive colors for the symbol interpretations. For example, we avoided using red, which could be interpreted as flagging up an error, and green, which could be interpreted as signalling a correct interpretation.

5. How Can We Facilitate Error Correction?

Once users have pinpointed errors, they then have to memorize them until they reach the end of the process. However, the gradual forgetting of visual patterns is amplified when a visual distractor prevents rehearsal [8]. We asked 36 volunteers to check the outcome of an automatic recognition process and circle any errors they found. Half of them could interrupt the recognition process and circle the errors as and when they spotted them (see **Figure 3**). The other half had to wait until the end of the process to do so.

Results showed that participants who interrupted the system finished the task significantly more quickly than the others. Participants who could not interrupt the system ended up having to check the interpretation a second time afterwards, because they had forgotten some of the errors. Regarding the "provide shortcut" heuristic [7], participants had two ways of correcting errors: an intuitive pause button at the top of the screen and a direct click on the error (invisible to novices).

6. Can the System Support Users?

We assessed the functionality whereby the software asks the user for help whenever a symbol proves difficult to interpret. Forty-eight participants were asked to supervise the automatic recognition of plans. For twenty-four of them, the software might stop at any time to ask the user if the most recent interpretation was relevant or not (see **Figure 4**). If not, it proposed possible corrections. The other twenty-four participants were not asked for their input.

Results showed that this type of questioning can save time, and errors signalled in this way are more likely to be corrected. However, we must be careful with this functionality, as there is a risk that users may relax their vigilance.

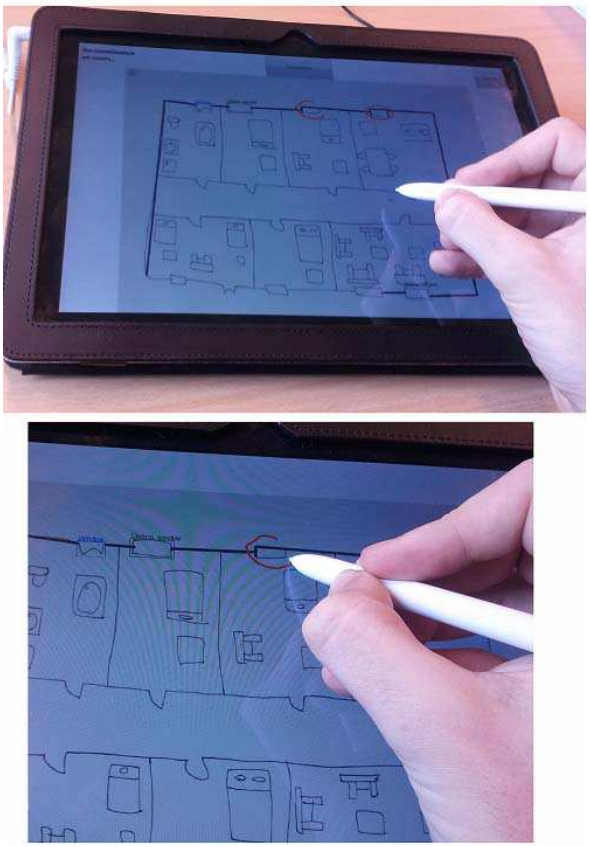

Figure 3. Participant circling an error before the end of the automatic recognition process.

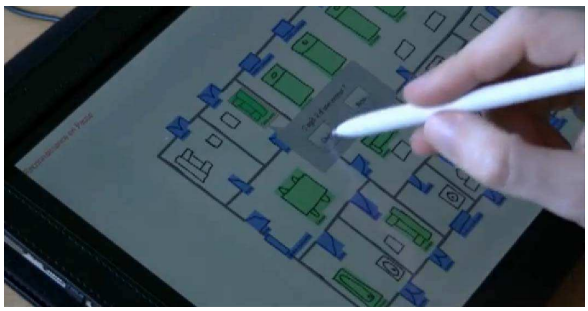

Figure 4. The system asks the user for help.

7. From Technocentric Design to User-Centred Design

Results showed how an anthropocentric approach based on user-centred design, consisting of experimental tests and usability heuristics, makes it possible to specify the functions and properties of automatic recognition software. At the end of the technological design process, just 33% of users corrected all the errors contained in simple plans (*i.e.* 15 symbols), whereas at the end of anthropocentric design process, 75% of users corrected all the errors contained in complex plans (*i.e.* 60 symbols).

In future works, requests to users must be evaluated more precisely. The benefit of this type of assistance seems to be largely dependent upon its accuracy. Consequently, several degree of accuracy must be tested in further experiments.

REFERENCES

[1] J. Llados, E. Valveny, G. Sanchez and E. Marti, "Symbol Recognition: Current Advances and Perspectives," In: *Graphics Recognition. Algorithms and Applications*, *Vol.* 2390 *of Lecture Notes in Computer Science*, Kingston, Springer Berlin Heidelberg, 2002, pp. 104-128.

[2] A. Ghorbel, A. Lemaitre and E. Anquetil, "Competitive Hybrid Exploration for Off-Line Sketches Structure Recognition," *International Conference on Frontiers in Handwriting Recognition* (*ICFHR*), Bari, 18-20 Septembr 2012, pp. 571-576.

[3] A. Wilson, M. Bekker, H. Johnson and P. Johnson, "Costs and Benefits of User Involvement in Design: Practitionerss' Views," Human-Computer Interaction, London, 1996.

[4] ISO 9241-210, "Ergonomics of Human-System Interaction-Part 210: Human-Centred Design for Interactive Systems," ISO, 2010.

[5] R. E. Mayer, "Multimedia Learning," Cambridge University Press, New York, 2001.

[6] C. J. H. Ludwig, A. Ranson and I. D. Gilchrist, "Oculomotor Capture by Transient Events: A Comparison of Abrupt Onsets, Offsets, Motion, and Flicker," *Journal of Vision*, Vol. 8, No. 114, 2008, pp. 1-16.

[7] R. Molich and J. Nielsen, "Improving a Human-Computer Dialogue," *Communication of the ACM*, Vol. 33, No. 3, 1990.

[8] D. A. Washburn and R. S. Astur, "Nonverbal Working Memory of Humans and Monkeys: Rehearsal in the Sketchpad?" *Memory & Cognition*, Vol. 26, No. 12, 1998, pp. 277-286.

