User Interface Design by Sketching: A Complexity Analysis of Widget Representations

Suzanne Kieffer, Adrien Coyette, Jean Vanderdonckt
Université catholique de Louvain, B-1348 Louvain-la-Neuve (Belgium)
{suzanne.kieffer, adrien.coyette, jean.vanderdonckt}@uclouvain.be

ABSTRACT
User interface design by sketching, as well as other sketching activities, typically involves sketching objects through representations that should combine meaningfulness for the end users and easiness for the recognition engines. To investigate this relationship, a multi-platform user interface design tool has been developed that enables designers to sketch design ideas in multiple levels of fidelity with multi-stroke gestures supporting widget representations and commands. A usability analysis of these activities, as they are submitted to a recognition engine, suggests that the level of fidelity, the amount of constraints imposed on the representations, and the visual difference of representations positively impact the sketching activity as a whole. Implications for further sketch representations in user interface design and beyond are provided based on usability guidelines.

Author Keywords
Level of fidelity, shape recognition, sketching, user interface design, user interface prototyping.

General Terms
Design, Human Factors, Languages.

ACM Classification Keywords
D.2.2 [**Software Engineering**]: Design Tools and Techniques – *User interfaces*. H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *Graphical user interfaces*. I.3.6 [**Computer Graphics**]: Methodology and Techniques – *Interaction techniques.*

INTRODUCTION
Sketching is universally recognized for its natural [9], unconstrained [25], and informal [6] virtues in multiple areas of human activity, such as graphic design [1], layout design [11], visualization [26], and user interface (UI) design [7,19,22]. As long as the sketching is not submitted to a recognition engine, the end user does not perceive any shortcoming apart from little or no reusability of the sketches for future steps in the design process. When it comes to recognize what the end user has sketched, e.g. for beautification [1] or interpretation [3,8], the end user may feel again constrained as she knows that every gesture should be performed correctly to be properly recognized, thus diminishing the virtue of naturalness. Recognition engines thus face a dilemma when defining the representation of objects to be recognized: either the representations are close to the real world but hard to recognize or they are simplified to be recognized, but not meaningful for the user.

EICS'10, June 19–23, 2010, Berlin, Germany.

More specifically, UI design by sketching has already demonstrated several advantages: UI sketching is preferred over traditional interface builders, especially by end users [9,18] and could be performed at different levels of fidelity without loosing advantages [20,25]: the amount of usability problems discovered with a sketched design is not inferior to those corresponding to a genuine UI [24], the expressive power of a sketched UI remains the same [25], a sketched UI provides quantitative and qualitative results that are comparable to traditional UI prototypes except that the cost is reduced [21], UI sketching encourages exploratory design and fosters communication between stakeholders more than any other prototypes [23], flexibility is superior to UI builders [25], authoring tools [2], and paper prototypes [25].

There are a number of problems with traditional sketching methods of UI design that make these methods challenging for novice users and inefficient for expert users. The first problem is related to the meaningfulness of representations: what is the best object representation? Should multiple representations of the same object be offered? How should it be sketched? Should it be sketched in one stroke or several strokes? If a representation is not meaningful enough for an end user, the representation will be forgotten or badly drawn. The second problem is that the result of the chosen representation is often far from what is expected by the novice user and difficult to reproduce [15]. The third problem with traditional recognition engines is that the representations should be different enough [13] and sketched precisely enough to be efficiently recognized [1,3,11].

To address these problems, we developed a UI sketching tool that provides original functionalities with respect to state-of-the-art software, as described in the following section. These functionalities are detailed in the next section. The Representation Experiment section reports on the results obtained from a first experiment to identify the representations preferred by designers and end users. The Complexity Experiment section describes the experiment used to test how these representations are sketched in the UI sketching tool, includes results and discussion. A summary of the main contributions of this paper and a description of the planned follow-up work conclude the paper.

RELATED WORK

During the UI development life cycle, the design step is often characterized as a process that is intrinsically open (new designs may appear at any time that require further exploration), iterative (several cycles are performed to reach a solution), and incomplete (not all information is available at design time) [10,12]. The area of UI design by sketching has been extensively researched to identify appropriate techniques such as paper sketching, prototypes, mock-ups, diagrams [6,9,11,13,16]. Several software for UI design by sketching emerged from this research: DENIM [11], DEMAIS [2], EtchaPad [15], FreeForms [18], InkKit [5], JavaSketchIt [3], Satin [7], Silk [9,10], SketchiXML [4], and SketchRead [1], to name the most representative ones.

Since the needs of rapid UI prototyping vary depending on the project and allocated resources, it makes sense to rely on the level of fidelity. The level of fidelity expresses the similarity between the final UI and the prototyped UI. The UI prototype fidelity is said to be *high* if the prototype representation is the closest possible to the final UI, or almost in the same representation. This means that the prototype should be of high-fidelity in terms of presentation (what layout, what are the UI elements used), of global navigation and dialog (how to navigate between information spaces), of local navigation (how to navigate within an information space). More precisely, McCurdy *et al.* [14] identified five independent dimensions along which the level of fidelity could be more rigorously defined: the level of visual refinement, the breadth of functionality, the depth of functionality, the richness of interactivity, and the richness of the data model. In the remainder of this paper, the four first dimensions will be considered, the last one requiring a connection to a data model containing data samples.

Similarly to the above definition, the level of fidelity is said to be *low* if the prototype representation only partially evokes the final UI without representing it in full details. Between high-fidelity (Hi-Fi) and low-fidelity (Lo-Fi) [20] exists medium-fidelity (Me-Fi) [4]. We usually observe that a UI prototype only involves one representation type, i.e. one fidelity level. But due to the variety of stakeholders' input, several levels of fidelities could be combined together, thus leading to the concept of *mixed-fidelity*, where several different fidelities are mixed in the same UI design [14]. Beyond mixed-fidelity, *multi-fidelity* [4] is reached when a prototype simultaneously involves objects belonging to different levels of fidelity, but only one level of fidelity is acted upon at a time, thus assuming that a transition is always possible for an object from one level of fidelity to another.

THE SKETCHING TOOL USED IN THE EXPERIMENTS

In this section, we describe the UI sketching tool that will be the subject of the two next experiments by showing how it is different from state-of-the-art software. This sketching tool today consists of about 112,000 lines of Java 1.5 code and can be freely downloaded from www.anonymous.org, both the executable software and its full source code. This tool enables UI designers to sketch a UI as easily as on paper, while combining advantages of computer-based design [25]. At any time, the designer may ask the tool to recognize the UI being sketched and generate a running UI from these sketches. At any time, it also offers the following facilities that are detailed in the following sub-sections:

Object recognition. An object recognition engine recognizes and interprets 32 different types of widgets (ranging from check boxes and spin button to search buttons, progress bar, calendar, and video input), 8 basic predefined shapes (i.e., triangle, rectangle, cross, line, wavy line, arrow, ellipse, and circle), and 6 basic commands (i.e., undo, redo, copy, paste, cut, new window). This amount of recognized objects is superior to what can be found in other software equipped with a recognition engine in the same domain [3,4,9,19,18].

Each object is rigorously defined in terms of constituent shapes (any of the 8 aforementioned basic shapes) and constraints between them. Each constraint should belong to the set of the 31 constraints supported today:

areParallel, cross, hasInside, hasInsideInLowerRightCorner, hasInsideInTheCenter, hasInsideInTop, hasInsideInUpperRightCorner, hasInsideOnTheLeft, hasInsideOnTheRight, hasPositiveSlope, intersect, isCrossedBy, isHorizontal, isInside, isInsideInBottom, isInsideInLowerRightCorner, isInsideInTheCenter, isInsideInTop, isInsideInUpperRightCorner, isInsideOnTheLeft, isInsideOnTheRight, isOnTheLeftOf, isOnTheRightOf, isOnUpperLeftCorner, isSmall, isSquare, isThin, isUnder, isVertical.

Any object representation is expressed in a XML format stored in a graphical grammar [1,3] that is parsed and interpreted at run-time [8]. In this way, any custom object could be easily added by adding a new representation in the grammar. Each UI element can be sketched and recognized or not depending on its shape and the wish for the user to see it recognized or not. The object recognition is only on-demand. Those shapes which are not recognized are simply added and maintained throughout the process. Fig. 1 shows a UI design session where some UI objects have been sketched in Lo-Fi mode. In this mode, objects that are correctly recognized are beautified and the name is added. If an object is not recognized, it is simply maintained as it is, but could be annotated for further handling in the future.

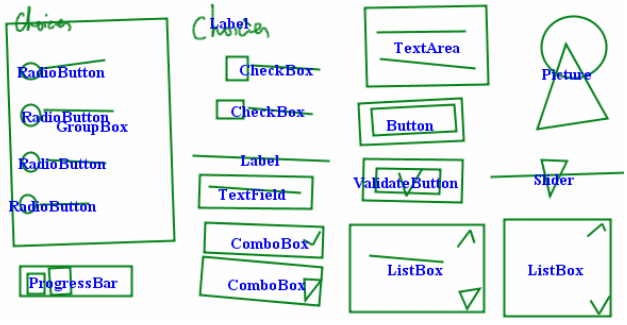


Figure 1. A typical UI design session with sketched objects.

Multiple object representations. Existing software incorporating an object recognition engine typically support only one single representation per object, most frequently through a mono-directional single-stroke gesture [3,15,18]. Our tool accommodates several representations for a single object, without affecting significantly the system response time. Fig. 2 reproduces an excerpt of the radio button representation. In addition, thanks to this logical definition, each representation could be sketched in a multi-stroke manner that is independent of the direction. In this way, left-handed or right-handed persons are equally supported.

```
- <widget type="RadioButton">
- <representation id="0">
  <constraint id="0" shape1="Line_1"
  shape2="Circle_0" condition="isOnTheRightOf"
  />
  <constraint id="1" shape1="Line_1" shape2="-"
  condition="isHorizontal" />
  <shape id="Circle_0" type="Circle" />
  <shape id="Line_1" type="Line" />
  </representation>
  …
</widget>
```

Figure 2. A representation for the radio button.

Multi-fidelity representation. Thanks to the object recognition process, the designer can input any UI object in any level of fidelity and see the result in any other level as the interpretation is immediate. In the same way, any custom object could be drawn in Lo-Fi and a predefined widget could be added in Me-Fi or Hi-Fi. Therefore, four fidelity levels are supported as recommended by [14]: none (only the drawing is displayed), Lo-Fi (the drawing is displayed with recognized portions), Me-Fi (the drawing is beautified where portions are recognized, including for basic shapes), and Hi-Fi (a genuine UI is produced with widgets for those recognized portions). Fig. 3 exemplifies multi-fidelity representations for a subset of the 27 widgets supported. To our knowledge, no existing software today supports so many widgets in different levels of fidelity as we have here.

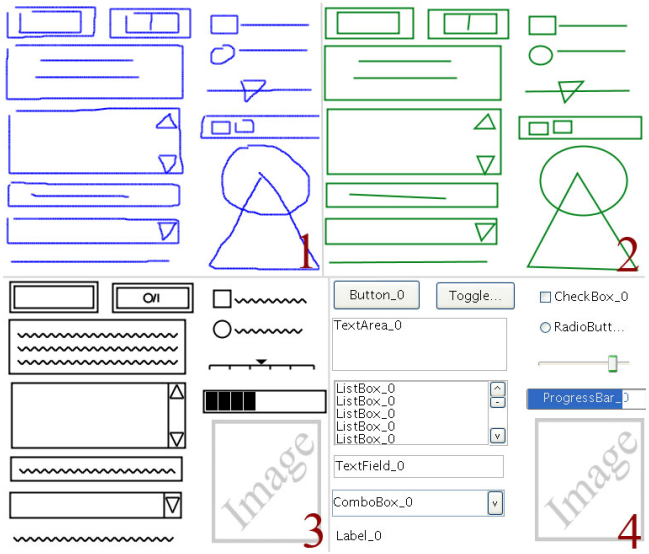


Figure 3. A set of widgets with the representations corresponding to the four levels of fidelities (i.e., none, low, medium, and high).

Fidelity transition. A slider allows the designer to easily switch between the four levels of fidelity (Fig. 3). Fig. 2-3 shows the representation after the designer moved from Lo-Fi to Lo-Fi, a mode in which only a rough, yet identifiable, object representation is produced that is often referred to as a wireframe representation. This representation is platform agnostic: it does not produce any representation that would suggest any particular window manager or UI builder. If the designer wants to obtain a Hi-Fi representation, then the slider may be switched to the last position (Fig. 2-4): Hi-Fi mode without the labels indicating the object types is displayed. In this case, the representation is made up of genuine widgets belonging to the widget set of the currently being used platform, here a Java platform. Different widget sets and look & feel could be used alternatively that mimic a Hi-Fi representation in other window managers and operating systems like Linux, Open Look, and MacOS X. If a UI element has not been recognized, it is simply kept as it is. For instance, if a histogram would have been sketched, it would not be altered so as to respect the naturalness of the design process as recommended in [14,16].

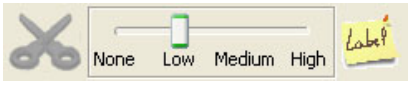


Figure 3. Slide to switch between levels of fidelity.

Gesture recognition. Sketching tool users sometimes complained that they are forced to learn a graphical representation [11,12] for every widget, shape or command. In order to support this user flexibility, each such object could of course give rise to a new representation in the graphical grammar. Some user studies revealed the need for the user to interactively define her own representations [6,11]. For this purpose, a gesture recognition system has been implemented based on hand gesture decomposition in order to customize the representation of all widgets, shapes, and commands according to each user's preferences (Fig. 4). One or several occurrences of a new gesture could be graphical defined that will then serve as a redundant input technique for every widget, shape, or command.

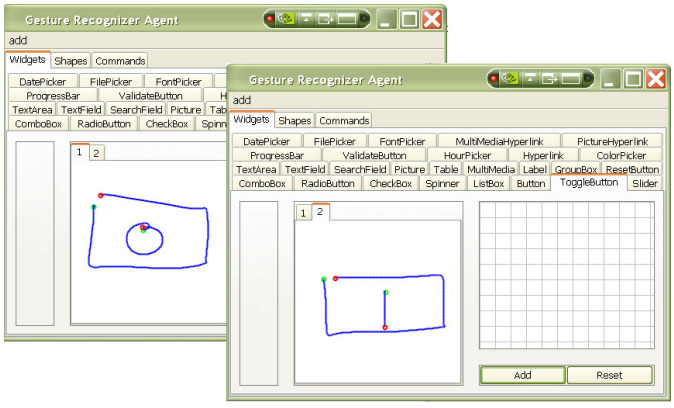


Figure 4. A graphical editor for a new object representation (a) and a gesture recognition system (b) where new gestures replace predefined objects (here, a gesture is drawn, added, and activated to represent a toggle button in a custom way).

Multiple output formats. At any time, the tool produces UI specifications in terms of a User Interface Description Language (UIDL) instead of UI code, which is the prevalent approach of most tools [3,9,11,18], but not all [5]. As opposed to many tools where little or no portions of the sketch could be reused, our tool always maintains up-to-date UI specifications, including the description of custom objects. It is also possible to define the navigation between these objects in the same way to address the second and third dimensions of McCurdy [14] (Fig. 5) as in [2,11].

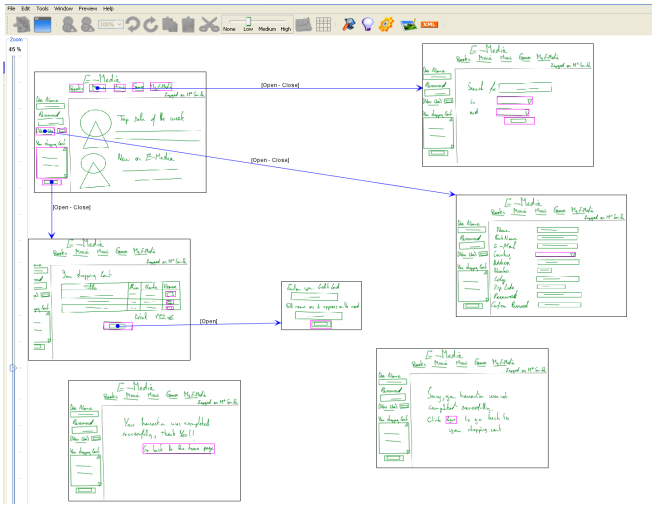


Figure 5. Definition of the navigation between UI objects.

Multi-platform UIs. By specifying project properties, the sketching tool enables designers to sketch UIs for a particular computing platform at a time or for several platforms in a coordinated way. It exports UI specifications in UIML (www.uiml.org), which is able to automatically generate code for HTML, Java, VoiceXML, and WML. As opposed to some tools which are dedicated to a particular environment (e.g., Visual Basic in FreeForms [18], Java in JavaSketchIt [3]), this tool is shipped with predefined profiles covering a wide range of different computing platforms. Each profile not only expresses constraints imposed by a particular platform (e.g., the screen resolution, a restricted widget set), but could also have a particular gesture data base for sketching those UI elements which are peculiar to this platform (e.g., a gesture associated to a histogram).

THE REPRESENTATION EXPERIMENT

The first experiment presented in this paper investigates which representation is mostly preferred and drawn depending on the user type (designer vs. end user).

Participants

Two groups of 30 subjects were randomly selected from a list of volunteer candidates: the first group was composed of people with relevant experience in computer science and UI design, while the second was composed of end users without any prior knowledge in UI design or computer science. The second group was also considered because the tool goal is to involve as much as possible the end user in the early prototyping process in order to bridge the gap between what they say and what the designer understands.

Methodology

A two phase analysis was carried out on both groups. The scope of the first part was to determine how members of each group would intuitively and freely sketch the widgets to be handled by the tool. From a cross-platform comparison of widgets, a widget catalogue was identified comprising the following 32 widgets: text, text field, text area, push button, search field, login, logout, reset form, validate, radio button, check box, combo box, image, multimedia area, layer, group box, table, separator, frame, hyperlink, anchor, list box, tabbed dialog box, menu, color picker, file picker, date picker, hour picker, toggle button, slider, progress bar, and spin button. Each widget was documented with its unique name, a screen shot and a small textual description (Table 1). Subjects were asked if they had ever seen each widget before and to provide a sketching representation.

Widget	Graphical presentation	Textual description
Search Field		This widget is composed of a text field and a button. It allows the users to submit a search.
Tabbed Dialog Box		This widget allows the user to switch from one pane to another thanks to the tab.

Table 1. Some objects submitted to the participants.

Then, from the widget representations provided during the first phase, we tried in a second phase, to extract the most common object representations. We grouped all these representations in categories with strong similarities as in [13]. Participants were then asked to rank the different representations according to their representativeness and preferences as a five-point Likert scale. On basis of these results we defined all the representations to be handled by the sketching tool. For instance, table 2 illustrates some representations.

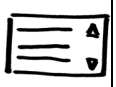
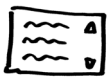
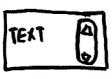
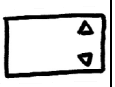
Representation 1	Representation 2	Representation 3	Representation 4	Representation 5
				

Table 2. List box representations submitted to the participants for the second part of the survey.

Results and Discussion

Based on the result distribution for each representation, we established the best representation with the following method. Firstly, we assessed whether any dependence exists between the participants. If this first step's results established a significant dependence, we then proceeded to the second phase and we computed the aggregate preference of both groups and the global preference. For each widget, the Kend-

all coefficient of concordance W test was computed. This coefficient expresses the degree of association among n variables, that is, the association between n sets of rankings. The degree of agreement among the 60 people who evaluated the representations is reflected by the degree of variation among the 6 sums of ranks. The comparison of the value obtained from this computation to the critical value shows that the null hypothesis (independence between participants) has to be rejected.

We can thus proceed to the second phase of the analysis and establish a ranking among all representations using the Borda Count method. The principle of the Borda Count Method is that, each candidate gets 1 point for each last-place vote received, 2 points for every next-to-last-place vote, etc., all the way up to N points for each first-place vote where N is the number of candidates. On the basis of this analysis we observed that both groups had almost the same preferences among the representations. Most of the time, the set of well considered representations is the same even if small changes in the sequence occur. Out of this results set, we considered the preferred representations with respect to their intrinsic complexity as explained earlier. For instance, list box 4 in Table 2 obtained a good score compared to the other representations, but its intrinsic complexity is very high as it requires hand writing recognition, which was not supported at the moment. Representations 4 and 5 in Table 2 were thus discarded from the final selection. Often, the set of representations selected for the list box is composed of the three first representations depicted in the corresponding set of representations.

The resulting catalogue of objects obtained from this study has then been submitted to the second experiment within the UI sketching tool. It is accessible at www.anonym.org.

THE COMPLEXITY EXPERIMENT

The second experiment presented in this paper investigates the effect of widget representation in the specific context of UI design by sketching. It investigates the potential influence of the level of fidelity as well, in order to strengthen the following result: users do not change their sketching strategy whatever the fidelity level is (see hypothesis A). Indeed, the usability study presented in [4] showed no significant impact of the "fidelity level" parameter on the user performances

The usability evaluation purpose is the quantitative analysis of user performance while they are in the interactive situation of sketching widget representations in the different levels of fidelity available in the sketching tool. Consequently, the evaluation criteria chosen to elicit the impact of both widget representation and level of fidelity on user performance are speed and accuracy. On the one hand, speed is representative of users' efficiency. On the other hand, accuracy is representative of users' effectiveness. The goal of the usability evaluation presented here is the validation of two hypotheses A and B:

Hypothesis A: user performance depends on the level of fidelity in which users sketch shapes. In other words, differences on users' speed (efficiency) and accuracy (effectiveness) should appear between the levels of fidelity, amongst none, low, medium, and high.

Hypothesis B: user performance depends on widget representation complexity. In other words, differences on users' speed (efficiency) and accuracy (effectiveness) should appear, these differences being function of widget representation properties, such as number, orientation, inclusion, intersection, juxtaposition, and sequence of atomic components.

Complexity characterization of representations

Prior to conducting the experiment, it is important to characterize the complexity of the widget representations to be used. In the tool studied, widget sketching equals: construction of basic shapes among circle, line, rectangle, and triangle, with respect to binary properties or constraints such as orientation, inclusion, or sequence of components. One specific combination of shapes and binary properties describes the gesture representation of a widget.

Gesture representation	Number of components	Specific orientation	Simple inclusion	Complex inclusion	Juxtaposition	Intersection	Sequence
Button	2			X			
Checkbox	2	X			X		
Combobox	2			X			
Label	1	X					
List box	5	X		X			X
Picture	2					X	
Progress bar	3	X		X			X
Radio button	2	X			X		
Slider	2	X				X	
Text area	3	X	X				X
Text field	2	X	X				
Toggle button	3	X		X			

Table 3. Complexity characterization of widget representations.

Table 3 presents some well-known widgets and, for each widget, a characterization of its gesture representation or visual code (first column). The characterization of a widget includes the number of shapes (from 1 for the label to 5 for the list box) and a combination of binary properties such as: specific orientation (vertical vs. horizontal), simple or complex inclusion, juxtaposition, intersection and sequence of components. This characterization of widgets is built upon Ware's visual grammar of diagram elements (node-link diagrams) [26].

Widgets in Table 3 are sorted according to the alphabetical order: no complexity order was introduced into the characterization at this step of the research. But it appears obvious that the complexity of the widget sketching task relies on the complexity associated to the visual code of the shape to sketch. From this observation, sketching a label, represented as a line, would be easier than sketching a combo box. Not only, the number of shapes and constraints vary between the

label and the combo box, but the kind of constraints to be used in a combo box (complex inclusion) is harder to satisfy than in a label (horizontal orientation).

Overall Experimental Design

Participants

Eleven volunteers participated to this experimental study, 5 females and 6 males. This group of participants was composed of experienced computer users, aged between 22 to 28 years. Moreover, all the participants were considered as expert pen users, as they had significant past experience with pen-based interaction.

Apparatus and experimental task environment

The computer system used in this study was a PC Dell Latitude D820 equipped with an Intel Core 2 Duo T7200 (2.0 GHz, 4 Mo cache level 2 memory) processor and 2 Gb of RAM memory. Participants were seated in front of a 21-inch Wacom Cintiq 21UX touch screen flat panel (Fig. 6) connected to this computer running the sketching tool described in the third section. This platform has been selected because it offers the best compromise between stylus precision and interaction surface. Screen resolution was set to 1,600 x 1,200 pixels, with a 32-bit color palette. The keyboard was not required to complete the task since the participants were supposed to use a stylus for sketching.

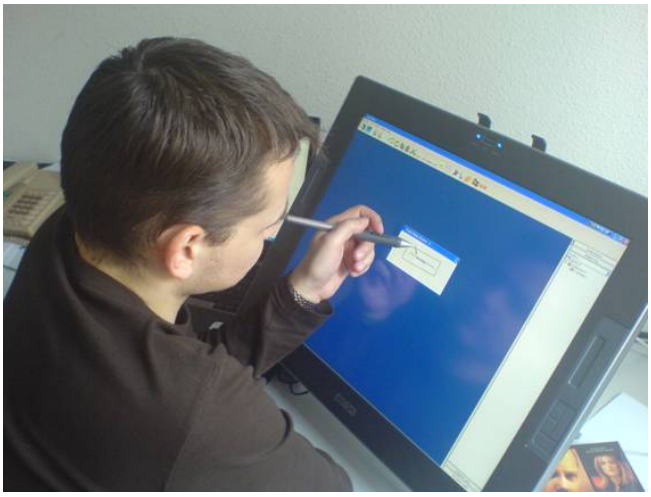


Figure 6. A participant performing the sketching tasks.

Task and procedure

Each participant received a detailed explanation of the research study. Following the short introduction to the test procedure and test purpose, they performed some training with the tool. Following the training session, participants performed the series of widget sketches with a constant rotation between the widget to be sketched and the fidelity level to be used. Simultaneously, all the relevant data were stored in log file so as to be used for statistical analysis.

Measures

The dependent variables used to assess the participant task performances were the widget sketching time (i.e., time in milliseconds until the widget sketching is effective), and the accuracy (i.e., number of delete operations until the widget sketching is effective).

Setup

The survey was based on a 4x12x2 factorial design; 4 fidelity levels were evaluated (none, low, medium and high), 12 frequently used widgets selected from the complete set resulting from the previous experiment (Table 3) and each widget was repeated twice for each level of fidelity. So, all participants received exactly the same 96 triplets (fidelity, widget, iteration) to sketch. However, the presentation sequence of these 96 triplets was randomized so as to neutralize potential task learning effects. The main directive for the participants was to sketch each triplet <widget, fidelity level, iteration>, as fast and precisely as they could. Participants were asked by a dialog box to sketch a given widget at a time. The fidelity level was automatically set and could not be changed by the user. Once the user considered the widget to be sketched, he had to click on one of the lateral buttons of the tablet PC to move to the next widget. If the widget to draw was present on the drawing surface, then the surface was cleared and a new widget was proposed to the participant. Otherwise, the participant was asked to finish the current component.

Results and discussion

The eleven participants completed the 96 timed trials each, for a total of 1056 trials. Hypothesis A is not validated by the experimental results whereas hypothesis B is strongly supported by the results of the experiment. The quantitative evaluation presented below relies on the statistical analysis of two measures: the widget sketching time (ST) in milliseconds and the number of delete operations (DEL).

Outliers removal

The data for 3 subjects (288 trials out of the 1056 trials), were removed as outliers, sketching time (ST) being greater than four standard deviations from the sketching time mean trial completion time. These outliers were not correlated with any of the participants. In addition, the data for the label (64 trials out of the 768 remaining) were also removed as outliers, the label being correlated with any of the other widgets in terms of sketching time (ST) distribution. Thus, the remainder of the analysis was performed using 703 trials, one data missing because of a technical problem.

One-Way ANOVA Procedure

The results of the One-Way ANOVA Procedure are presented in Table 4. Factors are fidelity level and widget representation. Responses are the widget sketching time in milliseconds (ST) and the number of delete operations (DEL). Significant influences are underlined. Results presented in Table 4 show that the level of fidelity is not a significant factor: neither on ST ($F=1.6813$, $p=0.1697$), nor on DEL ($F=1.9900$, $p=0.1141$). This result suggests that the level of fidelity selected to perform widget sketching tasks has no significant effect on users performance and, consequently, invalidates hypothesis A. Moreover, this can be interpreted as follows: users do not change their sketching strategy

whatever the fidelity level is, which strengthens the results presented in [4].

Factor	DF	ST (ms)	DEL
Fidelity level	3	F=1.6813 p=0.1697	F=1.9900 p=0.1141
Widget representation	7	F=7.4317 p<0.0001	F=2.9151 p=0.0014

Table 4. One-way ANOVA Procedure. Factors: fidelity level and widget representation. Variables: sketching times (ST) and delete operations (DEL).

On the other hand, results presented in Table 4 show that widget type is a significant factor: both on ST (F=7.4317, p<0.0001) and DEL (F=2.9151, p=0.0014). This result shows that widget representation has a significant effect –in the case of ST, one observes a highly significant effect of widget representation (p<0.0001)– on users performances and, consequently, validates hypothesis B. To summarize, in the specific context of widget sketching, users' efficiency and effectiveness in a context do not depend of the level of fidelity but depend on the widget representation. Next subsection will investigate further the influence of the widget representation, classifying the widgets according to both ST and DEL.

Widget classification

Regarding the results from the One-Way ANOVA Procedure above, we have computed complementary analysis on data by taking into account ST, first, and DEL, secondly. Indeed, widget representation has a highly significant impact on users speed, but "only" a significant impact on users' accuracy. So, the complementary analysis is built upon:

- First, a recursive partitioning (RP) of widget representation by ST in order to get groups of widgets (Fig. 7).
- Secondly, a sort within each group of widgets according to the widget recognition rate (Table 5). Recognition rates have been computed as error rates.

Recursive partitioning (RP) was applied to the dataset without outliers (703 trials) in order to elucidate statistically significant sub-groupings within the data by relating subjects' speed (ST) to the widget representation factor. The result of this process provides the decision tree presented figure 6 and it shows that widgets can be divided into 3 groups of widgets G1, G2 and G3:

- (G1) Text field, checkbox, radio button, picture;
- (G2) Slider, combo box, text area, button;
- (G3) Progressbar, listbox and togglebutton.

Then, a sorting according to the recognition rate (RR) of each widget was applied to each of the three groups. For instance, in the group 1, widgets are sorted by decreasing recognition rate: text field (98%), picture (97%), check box (95%), and finally radio button (94%). The same procedure was applied to each widget group. Recognition rates were computed as an error rate per widget. Results are presented in Table 5.

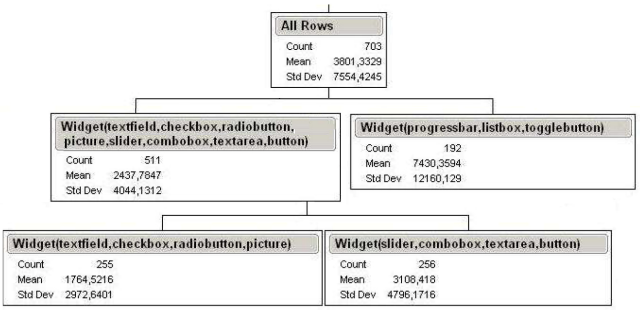


Figure 7. Recursive Partitioning on widget representation by sketching time.

Group	Widget	Number of components	Gesture representation	Recognition rate	Specific orientation	Simple inclusion	Complex inclusion	Juxtaposition	Intersection	Sequence
	Label	1		0,97	X					
1	Text field	2		0,98	X	X				
1	Picture	2		0,97					X	
1	Checkbox	2		0,95	X			X		
1	Radio button	2		0,94	X			X		
2	Button	2		0,95		X				
2	Text area	3		0,94	X	X				X
2	Slider	2		0,92	X				X	
2	Combobox	2		0,86			X			
3	List box	5		0,91	X		X			X
3	Progress bar	3		0,84	X		X			X
3	Toggle button	3		0,8	X		X			X

Table 5. Widget classification.

What is interesting in Table 5 is the homogeneousness of characteristics one can observe in groups 1 and 3 mainly. The group 1 includes the text field which is the most basic widget after the label, the picture, and the couple check box and radio button, both constructed from the combination of specific (horizontal) orientation and juxtaposition. The group 3 includes list box, progress bar and toggle button. Results for this group are homogenous in the sense that characteristics for each widget are exactly the same: specific orientation, complex inclusion and sequence. Moreover, the number of shapes is 3 (progress bar and toggle button) or 5 (list box).

These characteristics may explain the difference between users' performance: (ST=7430 ms, RR=0.85) for the group 3, compared to (ST=3108 ms, RR=0.9175) for the group 2 and (ST=1765 ms, RR=0.96) for the group 1.

Table 5 supports these observations, the last four elements (i.e., combo box, list box, progress bar and toggle button), are the widgets that required the more time with the highest error rate. We can observe that all of the four widgets are built using complex inclusion in addition to more simple graphical codes. Moreover, the ranking of the widget illustrate that a larger set of constraints tend to increase the recognition rate and the time required. The next section will

investigate the potential influence of such characteristics on user performance, by introducing in the statistical analysis the number of shapes as well as the binary properties of the widgets (Table 3) as factors. It is expected that these variables will have some significant influence on the sketching performance of the participants.

Widget characteristics

In order to deeply investigate the relative influence of each widget characteristic on user performances in a context of UI sketch, a One-Way ANOVA was applied to the data. The ANOVA procedure was computed on 703 trials (Table 6).

Characteristic	ST (ms)	DEL
Specific orientation	F=4.3378; p=0.0376	F=1.0632; p=0.3028
Simple inclusion	F=5.0825; p=0.0245	F=7.2306; p=0.0073
Complex inclusion	F=50.3739; **p<.0001**	F=24.3516; **p<.0001**
Juxtaposition	F=11.9572; p=0.0006	F=2.7937; p=0.0951
Intersection	F=6.4777; p=0.0111	F=1.6839; p=0.1948
Sequence	F=16.1455; **p<.0001**	F=2.5448; p=0.1111
Number of shapes	F=25.8457; **p<.0001**	F=7.1275; p=0.0009

Table 6. One-Way ANOVA Procedure. Factors: widgets characteristics. Variables: sketching times (ST) and delete operations (DEL). Highly significant results are in bold face.

Considering both ST and DEL, most relevant characteristics are complex inclusion, sequence and number of shapes (Table 6, cyan rows). Indeed, complex inclusion, sequence and number of shapes are highly significant factors. All factors have an impact on sketching times (Table 6, column ST: all factors are significant for sketching times), but only simple and complex inclusion and number of shapes do have a significant influence on delete operations (Table 6. column DEL: specific orientation, juxtaposition and intersection do not have a significant influence on delete operations). These statistical results suggest that complex widgets in terms of inclusion, sequence and number of shapes are slower and more difficult to sketch by users. Such characteristics –or "constraints", regarding widget representations implementation within sketch-based UI design tools– should be used carefully by programmers.

Label, text field, and picture are basic widget: between one and two shapes, high recognition rate (>0.97). The check box and the radio button can be considered as basic widgets as well, the only difference being the juxtaposition property. The other widgets seem to be more complex to sketch. Surprisingly, the intersection-based slider is "stuck" between inclusion-based widgets, all being sketched from a rectangle. This result may be explained by the small size of the triangle intersection the line in the slider. In addition, we were surprised from the toggle button bad results.

Why a toggle button is so hard compared to a push button? The answer may rest in the addition of the constraint of orientation of the (vertical) line inside the two rectangles of the toggle button. The same interpretation can be made about the combo box and the list box. The more constraints and shapes are added, the more complex the representation becomes to sketch.

Learning effects

To investigate the effects of task learning, we have computed mean and standard deviation by quartile on both ST and DEL. Results are presented in Table 7.

Quartile	Avg. ST (ms)	Std dev ST	Avg DEL	Std dev DEL
1	5055.92	8624.8	0.285714	0.863414
2	3768.70	10171.1	0.179775	0.896469
3	3245.88	4646.1	0.189655	0.740026
4	3136.02	5164.2	0.164773	0.821920

Table 7. Learning effects.

Results presented in Table 7 show that there is a learning effect of the task. First, both the sketching times (ST) and the number of delete operations (DEL) decrease with the time during the test (see Avg ST and Avg DEL from quartile 1 to quartile 4). Secondly, the standard deviation for both ST and DEL are, on the one hand, high during the first half of the test in comparison with the second half and, on the other hand, combined with high "finished" as well. These two observations typically indicate an effect of task learning. A One-way ANOVA Procedure was also combined with the computations above. It has revealed only a tendency about the eventual impact of the "quartile factor" on sketching times (F=2.3927, p=0.0674). Quartile is not a significant factor for number of delete operations (F=0.7587, p=0.5175). This result may be due to the fact that participants were all considered as designers, at least expert pen users.

CONCLUSION AND FUTURE WORK

Regarding to the UI sketching tool, we demonstrated that it is possible to come up with a tool combining the following facilities: multi-stroke bi-directional sketching of representations, object recognition based on a logical and extensible graphical grammar, ability to recognize multiple representations for a same object (either in a predefined way through the object recognizer or in a user-defined way through the gesture recognizer), multiple levels of fidelity and easy transition between them. This combination of facilities makes the UI sketching tool described in this paper unique. This tool has then been subject to two experiments: one for determining the most preferred representations for each object (other tools may benefit from this) and one for determining the influence of the level of fidelity. This lead us to several empirical conclusions.

Firstly, we have observed that the level of fidelity did not have any impact on the sketching of any individual widget. This is the most important conclusion because it is already

known that several levels of fidelity should be supported [14,20], but not that these various levels do not denaturize the essence of sketching. Naturally, such observation does not imply that a prototyping tool can choose to use indifferently any level of fidelity in isolation. The various levels of fidelity should be supported. Indeed, the level of fidelity is likely to influence the creation of a complete UI, as some representation may give an impression of almost finished results, as an example. Here, we only demonstrate that the time needed to build a given widget is not dependant of the level of fidelity to be used. Therefore, a UI design by sketching could be estimated equally good in performance independently from its level of fidelity.

Secondly, and unsurprisingly, we have also demonstrated that the quality of the recognition was significantly dependant of the type of widget representation. This observation is promising and rich as it provides valuable information for the development of any new graphical grammar, and for the improvement of some part of the application. We observed strong differences between the widget representations.

This lead to us to the following conclusion as a guideline: when defining a widget representation to be sketched, **a minimal amount of constraints should be involved**, especially when ambiguities between the representations are unlikely. For instance, the text field representation requires the enclosed line to be horizontal, but the line could be drawn with many other orientations for the same results as there are not any other representation composed of a single rectangle and line. This guideline could be generalized into the following one: any representation of an object to be sketched should minimize the amount of constraints whose types have been defined in Tables 3 and 5.

This conclusion also complements the results provided by the study reported in [13]: not only the visual difference should be well established between the representations to be sketched, but also they should minimize the amount of constraints required to sketch the object. Therefore, it is not only a matter of visual difference, but also a matter of sketching simplicity. The example of toggle button is revealing for this purpose.

The last significant observation made during this survey is related to the learning effect: we observed for all the participants that their overall performance was significantly higher at the end of the survey. They drew the widget more precisely, as the recognition rate is higher, in less time. Obviously, these two observations are related; the lower time at the beginning of the test can be partially attributed to the numerous delete operations.

Therefore, any UI design tool by sketching should address simultaneously the following requirements by decreasing order of importance:

- *Naturalness*: it is necessary that the UI objects being sketched are as natural as possible first. Then, the visual similarity should be considered to easily differentiate the various representations. And finally, the drawing constraints must be minimized in order to limit the exploring capability of the user. The results of such a UI prototyping process may be not immediately similar to a final interface, but the easy transition from one level of fidelity to another one is greatly appreciated. In the sketching tool described in this paper, the two input methods supported are the handwriting and the sketching by object recognition and gesture recognition. These two expressions means are well known for supporting highly creative design process [18,22].
- *Non-obtrusion*: it is necessary for the system supporting the sketching to be the less obtrusive as possible so as to avoid disturbing the designer during the prototyping phase. The low fidelity representation should not introduce new tasks or actions that are external to the original nature of the activity of prototyping.
- *Continuity*: the system supporting the sketching should support the drawing continuously whatever the nature of the object prototyped (e.g., an interaction object, a text, a drawing, or multimedia contents). The user should not have to change the mode of drawing if an object of different nature must be represented.
- *Recovery*: the effort provided for the sketch should be reused in the next step in the UI development life cycle of the interactive application. In theory, to minimize the costs, the effort supplied during this prototyping, whatever the level of fidelity is, should be recovered as much as possible in the continuation.

The above requirements could be turned into guidelines for a sketching tool, but may require further investigation in any particular domain where the object representation should first satisfy the conventions of the domain. The next experiment we will conduct is to see how these representations affect end users in the same way in the context of gesture annotation of medical images.

ACKNOWLEDGMENTS

The authors would like to warmly thank the anonymous reviewers for their constructive comments on an earlier version of this manuscript. In particular, we are very thankful for the English native speaker who made a tremendous job in reviewing the contents, but also the spelling and grammar. We gratefully acknowledge the support of the projects FP7 Human, FP7 Serenoa, and ITEA2 UsiXML (funded by European Commission and Région Wallonne).

REFERENCES

1. Alvarado, C. and Randall, D. SketchRead: a Multi-Domain Sketch Recognition Engine. In *Proc. of UIST'04*. ACM Press, New York (2004) pp. 23-32.
2. Bailey, B.P. and Konstan, J.A. Are informal tools better? Comparing DEMAIS, pencil and paper, and Authorware for early multimedia design. In *Proc. of CHI'03*. ACM Press, New York (2003), pp. 313-320.

3. Caetano, A., Goulart, N., Fonseca, M., and Jorge, J. JavaSketchIt: Issues in Sketching the Look of User Interfaces. In *Proc. of AAAI'02 Spring Symp. on Sketch Understanding*. AAAI Press, Menlo Park, pp. 9-14.

4. Coyette, A., Kieffer, S., and Vanderdonckt, J. Multi-Fidelity Prototyping of User Interfaces. In *Proc. of Interact'07*, Springer-Verlag (2007), pp. 149-162.

5. Chung, R., Mirica, P., and Plimmer, B. InkKit: a Generic Design Tool for the Tablet PC. In *Proc. of CHINZ'05*. ACM Press, New York (2005), pp. 29-30.

6. Hong, J.I., Li, F.C., Lin, J., and Landay, J.A. End-user Perceptions of Formal and Informal Representations of Web Sites. In *Extended Proc. of CHI'00*, pp. 385-386.

7. Hong, J.I. and Landay, J.A. Satin: a toolkit for informal ink-based applications. In *Proc. of UIST'00*, pp. 63-72.

8. Kara, L.B. and Stahovich, T.F. Hierarchical parsing and recognition of handsketched diagrams. In *Proc. of UIST'04*. ACM Press, New York (2004), pp. 13-22.

9. Landay, J.A. and Myers, B.A. Interactive Sketching for the Early Stages of User Interface Design. In *Proc. of CHI'95*. ACM Press, New York (1995), pp. 43-50.

10. Landay, J.A. and Myers, B.A. Sketching interfaces: toward more human interface design. *IEEE Computer 34*(3), 56-64.

11. Lin, J., Thomsen, M., and Landay, J.A. A visual language for sketching large and complex interactive designs. In *Proc. of CHI'02*, ACM Press, pp. 307-314.

12. Long, A.C., Landay, J.A., and Rowe, L.A. Implications for a gesture design tool. In *Proc. of CHI'99*, pp. 40-47.

13. Long, A.C., Landay, J.A., Rowe, L.A., and Michiels, J. Visual similarity of pen gestures. In *Proc. of CHI'00*. ACM Press, New York (2000), pp. 360-367.

14. McCurdy, M., Connors, C., Pyrzak, G., Kanefsky, B., and Vera, A. Breaking the Fidelity Barrier: An Examination of our Current Characterization of Prototypes and an Example of a Mixed-Fidelity Success. In *Proc. of CHI'06*. ACM Press, New York (2006), pp. 1233-1242.

15. Meyer, J. EtchaPad – Disposable Sketch Based Interfaces. In *Proc. of CHI'96*, ACM Press, pp. 195-198.

16. Newman, M.W. and Landay, J.A. Sitemaps, Storyboards, and Specifications: a Sketch of Web Site Design Practice. In *Proc. of DIS'00*. ACM Press, pp. 263-274.

17. Pajares, M., Ayala, P., Fajardo, I., Vicente, D., and Grana, M. Usability analysis of a pointing gesture interface. In *Proc. of IEEE Conf. on systems, man and cybernetics*. IEEE Computer Soc. Press, pp. 2652-2657.

18. Plimmer, B.E. and Apperley, M. Interacting with Sketched Interface Designs: an Evaluation Study. In *Extended Proc. of CHI'04*. ACM Press, pp. 1337-1340.

19. Rettig, M. Prototyping for tiny fingers. *Communications of the ACM 37*(4), 1994, pp. 21-27.

20. Rudd, J., Stern, K., and Isensee, S. Low vs. high-fidelity prototyping debate. *Interactions 3*(1), 1996, pp. 76-85.

21. Sefelin, R., Tscheligi, M., and Giller, V. Paper Prototyping – What is it Good for? A Comparison of Paper-and Computer-based Prototyping. In *Proc. of CHI'03*. ACM Press, New York (2003), pp. 778-779.

22. Snyder, C. Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces. Series in Interactive Technologies, Morgan Kaufmann, 2002.

23. Tohidi, M., Buxton, W., Baecker, R., and Sellen, A. User Sketches: a Quick, Inexpensive, and Effective Way to Elicit more Reflective User Feedback. In *Proc. of Nordi-CHI'06*. ACM Press, New York, pp. 105-114.

24. Virzi, R.A., Sokolov, J.L., and Karis, D. Usability problem identification using both low- and high-fidelity prototypes. In *Proc. of CHI'96*. ACM Press, pp. 236-243.

25. Walker, M., Takayama, L., and Landay, J. High-fidelity or Low-fidelity, Paper or Computer medium? In *Proc. of HFES'02*. HFES, Santa Monica (2002), pp. 661-665.

26. Ware, C. Information visualization: perception for design. Morgan Kauffman, San Francisco, CA, 1994.