Multi-Fidelity Prototyping of User Interfaces

Adrien Coyette, Suzanne Kieffer and Jean Vanderdonckt

Belgian Lab. of Computer-Human Interaction (BCHI), Information Systems Unit (ISYS)
Louvain School of Management (LSM), Université catholique de Louvain (UCL),
Place des Doyens 1, B–1348 Louvain-la-Neuve (Belgium)

Abstract. Multi-fidelity prototyping combines within a single user interface prototype several elements whose representations are reproduced with different levels of fidelity with respect to the final user interface: no fidelity at all, low fidelity, medium fidelity, and high fidelity. In order to effectively and efficiently support multi-fidelity, an original set of techniques is defined and discussed: multiple representation manipulation by sketching and gesture recognition, smooth transition between any representation at any time, prototype reusability, multi-platform support, and multiple user interface prototyping facilities. The full implementation of these techniques in prototyping software provides designers and developers with a unique environment for exploring multiple designs with unprecedented support for quickly designing interfaces from scratch or from previously existing design templates. An experimental study reveals that the multiple representation manipulation together with smooth transition represents a valuable advantage for naturally designing user interfaces. The prototyping software supports several aspects involved in the user interface development life cycle and is convenient for non-WIMP user interfaces.

1 Introduction

User-Centered Design (UCD) explicitly recommends in the User Interface (UI) development life cycle a specific stage where the UI could be prototyped based on the input of the future system's stakeholders: designers, developers, usability specialists, graphic experts, and end users. When the time comes to express and gather the user requirements, these stakeholders usually come to a design meeting with many ideas expressed in very different ways. Some prefer to convey their ideas through drawings, sketching, pictures, some others take screenshots of previously used interfaces to communicate representative examples, some others come without anything else than their past interaction experience and history, their own preferences. This therefore means that the prototyping stage should accommodate all these input types and integrate them into one single design. Since the stakeholders' inputs do not all come in the same format and with the same level of details, it is difficult to merge them in a straightforward way. With paper and pencil techniques [18], it is of course possible to manipulate all inputs on paper and to glue them so as to reach a unique UI, but its format remains largely inconsistent and almost not reusable for further design. When this preliminary design will be turned into more precise UI specifications, the quality of this representation does matter. Several tools have been invented to support UI design by sketching [1-4, 6, 8, 10, 13-16] since sketching probably represents the most natural way to convey ideas for the human being [1,2,18], but their predominant func-

tioning imposes some dedicated sketching activities that are then recognized (or not) and give rise to a working prototype (or not). None of them truly manipulate UI design artifacts with the aforementioned levels of details with the ability to easily switch from one representation to another.

In the remainder of this paper, Section 2 will define our understanding of the fidelity and how we generalize it into the concept of multi-fidelity. It will then compare state-of-the-art UI sketching tools against a series of seven criteria that will be further addressed throughout the paper. Section 3 will describe a series of techniques which, taken together, will allow our new sketching tool to satisfy the seven criteria. Section 4 will report on an experimental study where end users and UI designers evaluated the different levels of fidelity involved in the multi-fidelity paradigm. Section 5 argues that multi-fidelity could be equally used for other models (i.e., task, domain, abstract user interface) for a same UI or for other families of UI, such as physical UIs.

2 Related Work

Designing the right UI the first time is very unlikely to occur. Instead, UI design is recognized as a process that is intrinsically open (new considerations may appear at any time), iterative (several cycles are needed to reach an acceptable result), and incomplete (not all required considerations are available at design time) [8]. Consequently, means to support early UI design has been extensively researched to identify appropriate techniques such as paper sketching, prototypes, mock-ups, diagrams [18].

Since the needs of rapid UI prototyping vary depending on the project and allocated resources, it makes sense to rely on the notion of prototype fidelity. The *prototype fidelity* expresses the similarity between the final user interface (running in a particular technological space) 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 type. This means that the prototype should be 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). The 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 (low-fi) [17] exists medium-fidelity (me-fi) [9]. 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 fidelities could be imagined together, thus leading to the concept of *mixed-fidelity*, where several different fidelities are mixed in the same UI design [15]. As opposed to mixed-fidelity, we introduce the notion of *multi-fidelity* when a prototype may involve elements of different fidelities (like in mixed fidelity), but only one fidelity is acted upon at a time, thus assuming that a transition is always possible for an element from one fidelity to another for any element.

Prototyping software consequently falls into three categories depending on their fidelity level: *high-fidelity* tools support building a complete UI so that it can be directly executed and tested and as if the UI is a real one ; *medium-fidelity* tools support designing UI mock-ups giving more importance to the contents than the style with which these contents are presented ; *low-fidelity* tools focus more on the UI basic functionalities than on precise details through which these functionalities can be executed. Typical approaches found in lo-fi prototyping tools are the "paper and pencil

technique", the "whiteboard/blackboard and post-it approach" [19]. Such approaches provide access to most UI elements and prevent designers from being distracted from the primary UI design task. For instance, Berger [3] provides a predefined paper widget set for drawing a Microsoft Excel form which can then be turned into a true form.

UI designers who work out conceptual ideas on paper tend to iterate more and explore the design space more broadly, whereas designers using computer-based tools tend to take only one idea and work it out in detail [22]. The quality of the discussion among stakeholders is considered more fruitful with a hi-fi prototype than with a lo-fi mock up [18]. Lo-Fi prototyping, however, encourage the stakeholders to focus on the UI interaction rather than on details irrelevant at this level which do not influence the usability. Consequently, lo-fi prototyping offers clear advantages with respect to the hi-fi counterpart, but suffers from a lack of assistance and a lack of transition from lo-fi to hi-fi. On the one hand, maintaining an informal representation in lo-fi is observed to be important [13] so that stakeholders do not believe that the UI being designed is a final one, thus encouraging them to focus on design issues. On the other hand, once a lo-fi is finished, it is unclear how to proceed to a high fidelity level [23]. Me-fi comes in the game to "beautify" a lo-fi prototype without changing its functionality [9] and represents a possible evolution towards a final UI, but this transition is never supported in any software.

A recognized virtue of UI prototyping is its ability to extract usability problems so as to improve the UI design while prototyping [7]. The amount of usability problems extracted in a lo-fi prototype is not inferior to the amount of usability problems for a hi-fi prototype [22]. In addition, paper and computer media have been estimated equally valid for testing lo-fi, me-fi, and hi-fi prototypes [23]. In particular, computer media was considered more advantageous for automatic recording of user actions, for its ability to distribute and document the results of the UI prototype as opposed to paper [19,23]. Table 1 delivers the results of a comparative analysis where major prototyping tools are compared against seven criteria:

1. *Amount of fidelity*: most tools involve one or two fidelity levels (lo-fi and hi-fi), only one of them does support me-fi. When lo-fi is the single fidelity supported, it often means that this representation is converted into UI code afterwards (e.g., Visual Basic code for FreeForms, Java code for JavaSketchiIt, C code for Silk).
2. *Fidelity transition*: even less support a smooth transition between the fidelity levels at design time, even if we include easy transition to code for a final UI.
3. *Shape recognition*: a shape recognition algorithm is implemented in most tools in order to turn a sketched UI element into its real counterpart or to "beautify" it.
4. *Gesture recognition*: very few tools incorporate a gesture recognition algorithm to convert gestures into sketching commands or UI elements.
5. *Output reusability*: converting the sketched UI into a file which could be reused for the rest of the development life cycle is fundamental, but the output expressiveness and exploitability depend on the format (image vs. UI specifications).
6. *Multi-platform support*: still few tools do support UI prototyping for multiple computing platforms ranging from a desktop to a mobile platform.
7. *UI types*: all tools are tied up with a specific UI type (Graphical UI) and cannot be reused for other types of sketching activities or other UI types (e.g., non-WIMP).

The last line of Table 1 shows that our sketching tool is more advanced than the other tools with respect to all criteria, which will be demonstrated in the next section.

Table 1. Comparative analysis of prototyping tools by sketching.

	Amount of fidelities	Fidelity transition	Shape recognition	Gesture recognition	Output reusability	Multi-platform	UI types
Berger [3]	1 (hi-fi)				✓ (Excel VB code)		Excel form
Demais [2]	2 (lo-fi, hi-fi)		✓	✓	× (animation)		multimedia application
Denim [14]	1 (lo-fi)				× (image only)		Web UI
FreeForms [16]	1 (lo-fi)		✓		✓ (Basic code)		GUI
Ink-kit [6]	2 (lo-fi, hi-fi)	✓	✓		✓ (ASCII file)	✓	GUI
JavaSketch- It [4]	1 (lo-fi)	✓	✓		✓ (Java code)	✓	GUI
Meyer [13]	1 (me-fi)				× (image only)		GUI
Prototyper [15]	2 (lo-fi, hi-fi)				✓ (XML file)		GUI
Silk [10]	1 (lo-fi)		✓		✓ (C++ file)		GUI
SketchiXML [8]	2 (lo-fi, hi-fi)	✓	✓		✓ (UI specifications)	✓	GUI, PDA, mobile phone
Sketch-Read [1]	1 (lo-fi)		✓		× (image only)		GUI
Our tool	4 (no-fi, lo-fi, me-fi, hi-fi)	✓	✓	✓	✓ (UI specifications)	✓	Web UI, GUI, any type

3 Tool Support for Multi-Fidelity

The first step in our sketching tool consists of specifying parameters that will drive the prototyping process: the project name, the input device type (e.g., stylus, pen, mouse), the computing platform for which the UI is prototyped (a predefined profile exist for mobile phone, PDA, TabletPC, kiosk, ScreenPhone, laptop and a custom one could added). The user then enters into a UI design mode where any shape can be freely drawn and any text could be written. The tool is equipped with a series of facilities which taken together do support the multi-fidelity process as outlined before.

Shape recognition. A shape recognition engine is able to recognize and interpret 27 different types of widgets with the standard configuration (ranging from check boxes and spin button to search buttons, progress bar, calendar, video input), 8 basic 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). Each UI element can be sketched and be recognized or not depending on its shape and the wish for the user to see it recognized or not. The primary mode is lo-fi so as to create a context where the user feels free and unconstrained to draw any kind of shape, whether it can be recognized or not.

Those shapes which are not recognized are simply added and maintained throughout the process. Fig. 2 reproduces a typical session where a wide bunch of UI elements have been sketched in lo-fi mode. In this mode, elements which have been correctly recognized are beautified (the drawing is improved) and the name of the UI element has been added. Fig. 1 reproduces the lo-fi mode where the raw sketching was performed.

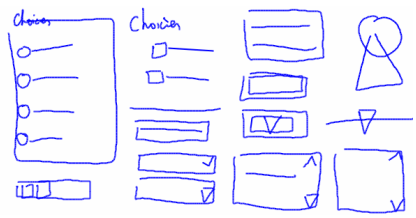

Figure 1. No-fi mode without labels.

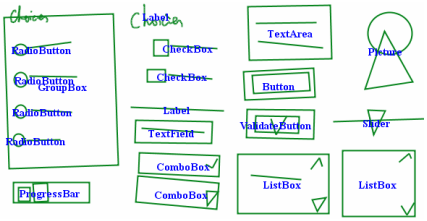

Figure 2. Lo-fi mode for sketching UI elements (with labels).

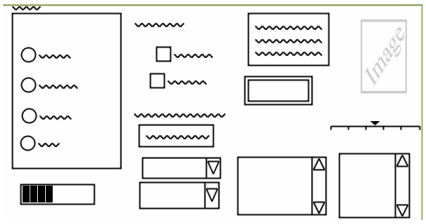

Figure 3. Me-fi mode without labels.

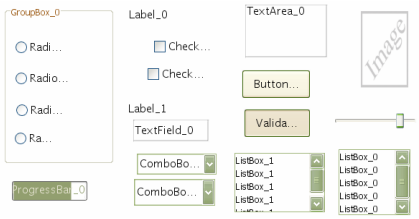

Figure 4. Hi-fi mode without labels.

Figure 5. Our software toolbar with fidelity level set on lo-fi.

Fidelity transition. A slider (Fig. 5) allows the user to easily switch between any fidelity level to another. Fig. 3 shows the representation after the user moved to me-fi, a mode in which only a rough, yet identifiable, element representation is produced. This representation is platform agnostic: it does not produce a representation which would suggest that a particular window manager, toolkit or environment has been selected. If the user really wants to obtain a hi-fi representation, then she may want to switch to the last position of the slider, which is demonstrated in Fig. 4: hi-fi mode without the labels indicating the elements types. In this case, the representation is made up of genuine widgets belonging to the widget set of the currently being used platform (different widget sets and look&feel could be used alternatively). A toggle button "Label" allows the user to display/undisplay the names of the recognized UI elements. 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 [14,17].

Amount of fidelities. Thanks to this process, the user can input any UI element in any fidelity level and see the result in any other level as the interpretation is immediate. In this way, a custom element 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: 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 true widgets for those portions corresponding to predefined widgets).

Gesture recognition. Sketching tool users may complain that they are forced to learn a graphical representation for every widget, shape or command. In order to sup-

port user flexibility, each such element is encoded in a graphical grammar of objects defined with logical relationships with variable degree of freedom. Fig. 5a shows how a multi-line edit field is graphically represented by a rectangle and two horizontal lines in it. In this way, the tool accommodates a larger variety of alternate representations for a same element. For this purpose, we based our implementation on an experimental study which reported the three most preferred representations for such UI elements [8]. Beyond this study, 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. 6b).

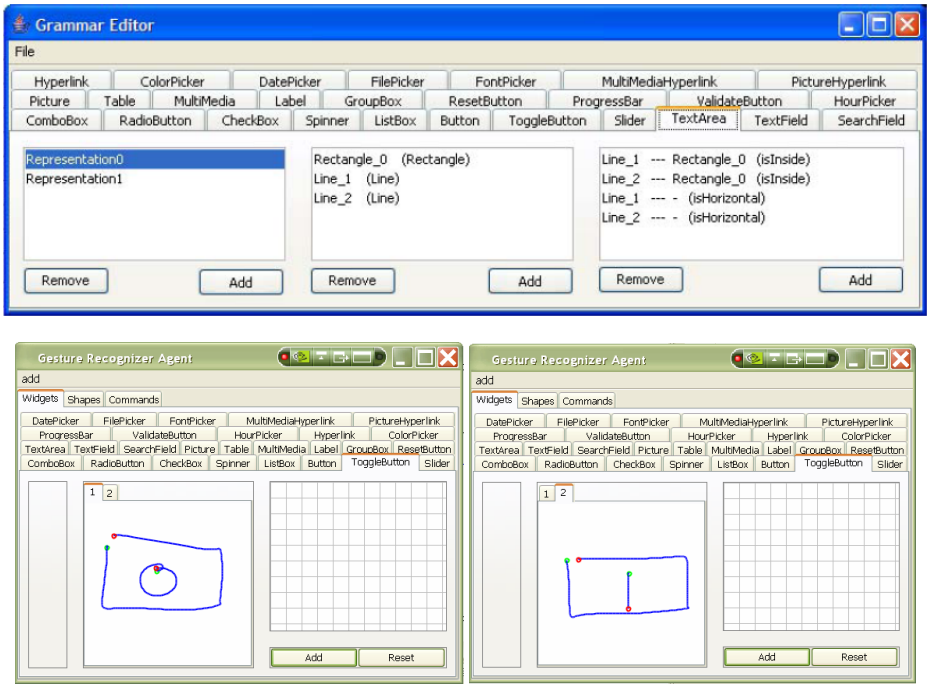

Figure 6. A grammar editor for a new representation (a) and a gesture recognition system (b) where new gestures replace UI elements (here, a gesture is drawn, added, and activated to represent a toggle button in a custom way).

Output reusability. At any time, our tool produces UI specifications in terms of a User Interface Description Language (UIDL). 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 widgets. It is also possible to define the navigation between these elements in the same way.

Multi-platform. The tool also exports UI specifications in UIML (www.uiml.org, which is able to automatically generate code for HTML, Java, VoiceXML, and WML) and UsiXML (www.usixml.org) [12,20]. As opposed to some tools which are dedicated to a particular environment (e.g., Visual Basic for FreeForms [17] or Java for JavaSketchIt [4]), our 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 above discussion shows that our tool satisfies the six first criteria highlighted in Table 1. The next section will investigate to what extent this tool supporting multi-fidelity is appreciated by end users and designers. Section 5 will then address the last criteria: it will exemplify how the tool could be used for other types of UI than merely web pages (like in DENIM [14]), GUIs (like in Prototyper [16] or in SketchiXML[8]).

4 Experimental Study on Fidelity Level

In order to evaluate how end users and UI designers appreciate the various fidelity levels at design time, an experimental study has been set up for investigating the effects of fidelity level on a UI design activity by sketching.

4.1 Method

Participants. Twelve volunteers participated in this study. Participants ranged in age from 23 to 39 years (M=30 years), including 6 females and 6 males to keep gender balance. Participants were selected on the basis of general inclusion criteria including age and profile (end user or UI designer). All participants were identified and recruited regarding their job in the computer science area (e.g., regular users, computer science researchers, developers, and UI designers from private companies). Table 2 summarizes the demographic information and the characteristics of the overall participant sample. Age represents the average number of years for the overall sample. Gender represents the frequency counts of males and females. General profile denotes the frequency in categories: end users vs. UI designers. Professional computer experience represents the average number of years for the overall sample while designing computer experience represents the average number of years for the UI designers only. The end users versus designers assessment was made in order to obtain a comprehensive profile of participants.

Table 2. Summary of participants' demographics and characteristics.

N	Age	Gender		Handedness	General profile		Computer experience	
		Male	Female		End users	User interface designers	Professional exp.	Designing exp.
12	30	6	6	Right	6	6	5.25	4

Table 3 summarizes the demographic and the characteristics of the participants based on the grouping. Age represents the average number of years for each participant group. Gender represents the frequency counts of males (M) and females (F) within each group. Professional computer experience represents the average number of years for each participant group. Designing computer experience represents the average number of years for the designers only.

Table 3. Summary of group profiles.

Group	N	Age	Gender	Professional exp/	Designing exp.
Designer	6	31	M=4, F=2	6.8	4
User	6	29	M=2, F=4	3.7	N/A

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 approximately 30 cm from a 21-inch Wacom Cintiq 21UX touch screen flat panel connected to this computer. 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. The sketching tool used in this experiment is the one whose implementation has been described in Section 3. The experimental task to be carried out by participants consists of designing two UIs (combined in a pair) in each of the following fidelity levels: Lo-Fi, Me-Fi, Hi-Fi, or No-Fi. Each UI contains eight widgets amongst the following alternatives: push button, check box, combo box, list box, progression bar, radio button, spinner, text area, and text field. A UI pair is considered to be complete once the eight widgets of both UIs have been entirely designed with the imposed fidelity level.

4.2 Protocol

Prior to experiment, participants were given an explanation of the research study and their role in the study. Following completion of the demographic questionnaire, the participants were briefed on how to use the setup and how to carry out the task. A short training period has been allocated for each participant to sketch a given UI pair until they feel confident in using the setup. They were also allowed switching between the four fidelity levels. The main part of the experiment consisted of designing four pairs of windows by sketching them in a pre-assigned fidelity level. The order of the four pairs of windows was randomly assigned. After these sketching tasks, participants were asked to complete a Computer System Usability Questionnaire (CSUQ) [11] and were interviewed according to a semi-structured scheme. The interview focused on their subjective satisfaction and perception about the study, the system and their preferences in term of fidelity level. The dependent variable used to assess participant task performance was Window Development Time (WDT), which represents the task duration (in seconds) required by a participant to design a window.

4.3 Results

Statistical analysis. One participant has not followed the instruction related to the order of the conditions. Consequently, the sample includes only 88 entries instead of 96. Due to the sample size, an analysis of variance (ANOVA) was used to examine the presence of significant differences in task performance, as measured by WDT. Table 3 reproduces the results of two analyses: influence of the fidelity level and influence of the user profile. The statistical significance is underlined.

Table 4. Tests for significant differences in performance.

ANOVA	Tests of Sig. Diff. Between groups
1) Fidelity (No/Lo/Me/Hi-Fi)-	F=1.8888; p=0.1377
2) User profile (User/Designer)	F=7.2719; p=0.0084

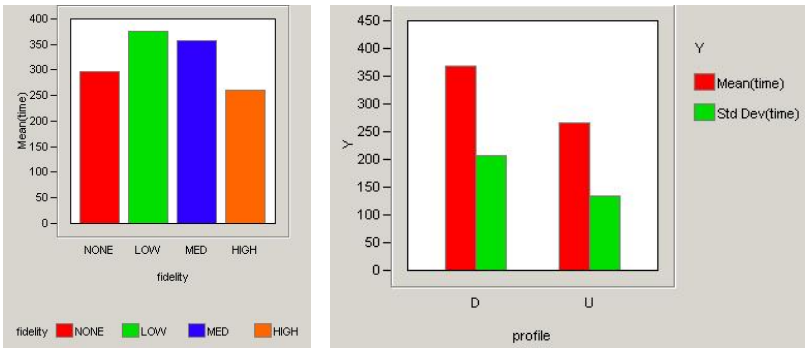

Figure 7. Mean WDT (seconds) for each fidelity condition, Mean WDT (seconds) for each participant group.

Although results from Table 4 show that the fidelity level had no influence on WDT, Hi-Fi demonstrated the fastest WDT (M= 261 seconds), respectively followed by No-Fi (M= 297 seconds), Me-Fi (M= 359 seconds) and Lo-Fi (M= 376 seconds) (Figure 6). In addition, the results from Table 3 show that user profile had a significant influence on WDT (F=7.2719; p=0.0084). Surprisingly, participants from the end users group demonstrated the fastest WDT compared to those from the designers group (respectively, M=267 seconds versus M=369 seconds – Fig. 7).

Table 5. Summary of overall sample CSUQ. Statistical indices are mean, median and standard deviation.

Subscale	Statistical Indices		
	Mean	Median	Std deviation
SYSUSE	4.04	4	1.52
INTERQUAL	5.39	6	1.14
OVERALL	4.83	5	1.17
INFOQUAL	4.45	4.5	1.37

Computer System Usability Questionnaire. The IBM CSUQ [11] is a public domain instrument to measure user satisfaction with computer system usability in the context of scenario based usability studies. The CSUQ is made up of four parts, each consisting of items ranked on a 7-point Likert scale: the overall satisfaction score (OVERALL: all 18 Items), the system usefulness score (SYSUSE: Items 1-8), the information quality score (INFOQUAL: Items 9-15), and the interface quality score (INTERQUAL: Items 16-18). This questionnaire has been chosen because of its acceptable reliability: a coefficient alpha exceeding .89 for all parts has been proved. Seven-point rating scales (1=totally disagree, 7=totally agree) were used because they allow three levels of either positive or negative ratings. Table 5 suggests that the system usefulness is moderately appreciated as well as the information quality (reasonably good mean, but large deviation). However, the interface quality and the overall user satisfaction are both assessed positively.

Subjective general comments and users preferences. Four of 12 participants judged the stylus uncomfortable because of a physical button located too close to their index finger. Four of 12 participants reported that some system functionality was not usable: the copy-paste was estimated too slow and required too many pointing gestures; the lack of drag-and-drop of sketched items was regretted since it is at the pre-

sent time replaced by the cut-paste functionality. Four on 12 participants considered that the speed of the recognition should be improved in the next version of the tool. In return, nine on 12 participants judged the tool as user-friendly and intuitive. This result is consistent with the INTERQUAL result reported above (Table 4). Moreover, eight on 12 participants considered the tool as fast and accurate in term of drawings/sketchings recognition. Finally, most of the participants reported a pronounced preference for Hi-Fi (5 participants on 12, including 2 designers and 3 users) and Me-Fi (5 participants on 12, including 3 designers and 2 users). They argued they felt "more comfortable" in those two levels because of the real-time interpretation of their drawings and the resulting UI aesthetics. Furthermore, 75% of the participants dislike the No-Fi (9 participants on 12, including 4 designers and 5 users). They claimed that this level "looks like a draft", which is consistent with [13].

Interpretation and discussion. The experimental task used in this study was a simplified version of a UI development life cycle. Time required by participants to develop UIs (WDTs) was used as an indicator on the usability of the fidelity levels. This metric revealed its shortcoming: WDT is not exact enough to be considered as representative of participant performance. Further usability studies need to include other metrics like the number of recognized/unrecognized shapes/texts/gestures, as well as the number of effective "widgets" that are added to the interface.

The statistical analysis revealed no significant impact of the "fidelity level" parameter on the user performances (speed). This result may be due to the fact that the level of fidelity has no influence on the sketching strategies adopted by the users, that is to say they perform the tasks in the same way, no matter what the level of fidelity is. In addition, the statistical analysis revealed a significant impact of the user profile (end user vs. designer) on the performances. Surprisingly, end users –with no experiment in interface design– are faster in performing the sketching tasks than the designers. This result may be due to the fact that designers do care a lot about the quality and aesthetics of the resulting interfaces (e.g., they systematically preserved alignment, symmetry, and semantic grouping of UI elements) compared to end users. Consequently, more time is required for designers to sketch valuable interfaces, regarding their own personal criteria. These results are consistent with some earlier findings [8].

Finally, the qualitative analysis revealed a pronounced user preference for both Hi-Fi and Me-Fi. This result suggests that participants, including both end users and designers, may prefer in terms of visual comfort, visual feedback, and widget recognition the fidelity levels that show a resemblance to the final UI. Differences observed between end users and designers are consistent with some other findings [2,8,22].

5 Multi-fidelity for Other User Interface Artifacts

In the previous experiment, multi-fidelity has been applied to the Concrete User Interface (CUI) level as defined in the Cameleon framework for a UI [5]. We show that our sketching tool can accommodate any UI type for any platform by choosing the right profile containing the constraints imposed by a particular platform. This profile influences the sketch recognition process as well as the trainable gesture recognition system. In this section, we show that the paradigm of multi-fidelity could be equally

used for other models involved in the UI development life cycle [3]: the task model [15], the domain model [6], and the abstract UI [20]. Each model consists of basic graphical elements which could be encoded in additional elements both in the graphical grammar and in the gesture recognition system.

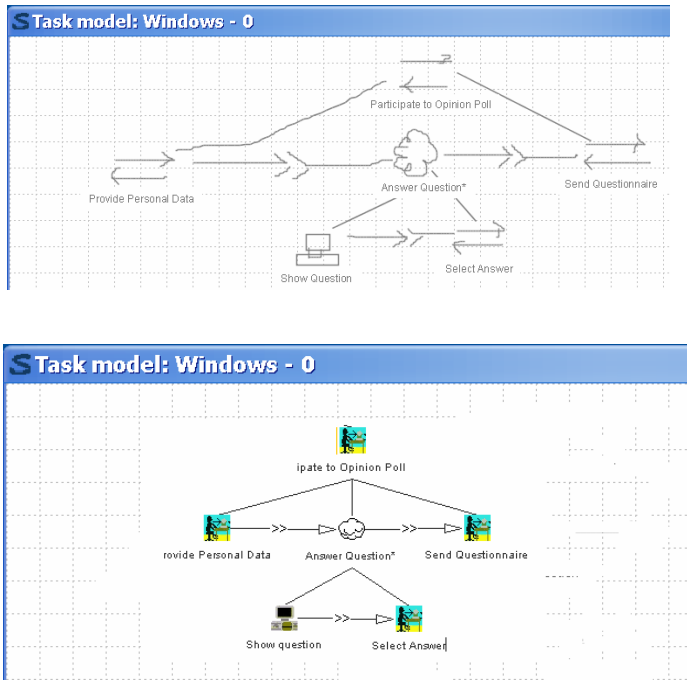

Figure 8. A task model sketched in lo-fi mode.

For instance, natural development of systems is fostered if a task model is drawn, e.g., on a drawing surface [15]. In our tool, a lo-fi approach could be adopted to sketch such a task model (Fig. 8) which could be straightforwardly recognized, interpreted and converted into a true task model (Fig. 8). In this way, it is possible to sketch all models involved in a particular UI development life cycle and link them together, which supports the principle of "sketching it all together". As long as a sketch could be decomposed into basic shapes such as rectangles, text (there is a ink-based recognition system for this purpose), lines, compound shapes, it is possible to sketch the representation in lo-fi and associate it to a beautification and a complete representation in hi-fi.

Then, we show that we could even sketch other families of UI provided we could imagine different representations belonging to different levels of fidelity. To go beyond the traditional paradigm of Graphical UIs, an example of a physical UI consisting of analogic and digital elements could be sketched similarly with the three fidelity levels. The output in this case consists of a description of a physical interface to be imported in the Pin&Play toolkit [21]. This toolkit allows developing physical interfaces by integrating software widgets and physical devices such as slider, toggle button, and potentiometer. Since the toggle button is not a standard element, is has been defined through a new custom gesture (Fig. 6), which could then be associated with the description of the genuine physical toggle button (such as a switch). Fig. 9 respec-

tively reproduces such a physical UI in lo-fi, me-fi, and hi-fi with smooth transition between these modes.

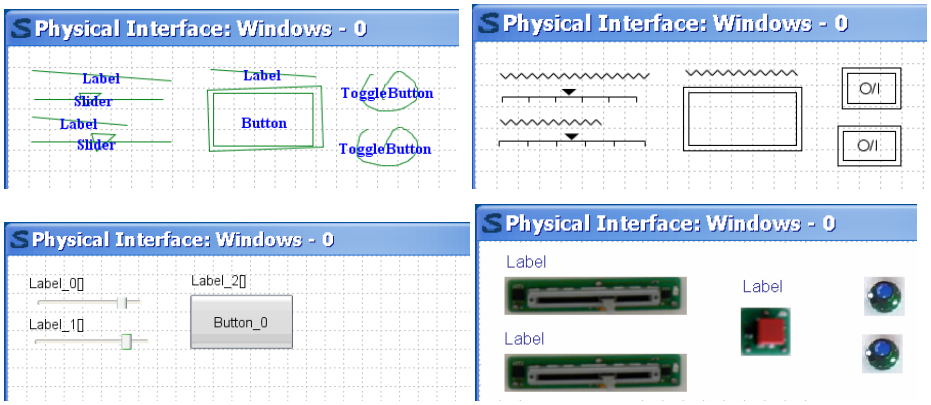

Figure 9. A simple task model recognized in hi-fi mode from its sketch in lo-fi mode.

5 Conclusion

As indicated in Table 1, our tool is superior to state-of-the-art prototyping tools by sketching in that it combines multi-fidelity with all criteria addressed simultaneously. The conducted experimental study revealed to what extent end users and designers do appreciate the freedom of design and the ability to smoothly progress from a UI design with moderate level of details (e.g., no-fi and lo-fi) to a more advanced level of details (e.g., me-fi and hi-fi). It is worth to note that the sketching facilities are equally appreciated by both end users (who are not necessarily designers) and professional UI designers. It is also particularly appreciated that, depending of the project evolution, any fidelity level could be switched to another one: not only for supporting the back and forth development life cycle, but also to incorporate UI elements which are expressed with different fidelity levels as they are provided by the stakeholders involved in the development team.

The combination of a shape recognition engine for predefined UI elements and a trainable gesture recognition engine allows the tool to be appreciated in many circumstances. The entire sketching tool described in this paper, along with its shape and gesture recognition systems for supporting multi-fidelity has been implemented in Java 1.5 and today consists of 45.000 lines of code. Our sketching tool can be freely downloaded from http://www.usixml.org/index.php?view=page&idpage=29 and its corresponding open source project.

The next development steps will consist in the development of an improved text detection algorithm. Indeed, we always try to proceed to a post treatment before trying to recognize a stroke. Detecting the text is far from being trivial and should be improved. We also plan to enhance overall performance of the application by optimizing some of the key algorithms.

And, finally, we will investigate to what extent the various modules of the software could accommodate other UI families, perhaps with other notations.

Acknowledgements

We gratefully acknowledge the support of the Request research project under the umbrella of the WIST (Wallonie Information Société Technologies) program under convention n°031/5592 RW REQUEST). We warmly thank J.A. Jorge, F.M.G. Pereira and A. Caetano for allowing us to use JavaSketchIt and the CALI library in our research. We gratefully acknowledge the support of the SIMILAR network of excellence (http://www. similar.cc), the European research task force creating human-machine interfaces similar to human-human communication of the European Sixth Framework Programme (FP6-2002-IST1-507609).

References

1. Alvarado, Ch., Randall, D.: *SketchREAD: A Multi-domain Sketch Recognition Engine*. In: Proc. of 17th Annual ACM Symposium on User Interface Software and Technology UIST'2004 (Santa Fe, October 24-27, 2004). ACM Press, New York (2004) 23–32.
2. Bailey, B.P., Konstan, J.A.: *Are Informal Tools Better? Comparing DEMAIS, Pencil and Paper, and Authorware for Early Multimedia Design*. In: Proc. of the ACM Conf. on Human Factors in Computing Systems CHI'2003 (Ft. Lauderdale, April 5-10, 2003). ACM Press, New York (2003) 313–320.
3. Berger, N.: *The Excel Story*. Interactions 13, 1 (January-February 2006) 14–17.
4. Caetano, A., Goulart, N., Fonseca, M., Jorge, J.: *JavaSketchIt: Issues in Sketching the Look of User Interfaces*. In: Proc. of the 2002 AAAI Spring Symposium - Sketch Understanding (Palo Alto, March 2002). AAAI Press, Menlo Park (2002) 9–14.
5. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J.: *A Unifying Reference Framework for Multi-Target User Interfaces*. Interacting with Computers 15, 3 (June 2003) 289–308.
6. Chung, R., Mirica, P., Plimmer, B.: *InkKit: A Generic Design Tool for the Tablet PC*. In: Proc. of 6th ACM SIGCHI New Zealand chapter's International Conference on Computer-Human Interaction CHINZ'2005 (Auckland, July 6-8, 2005). ACM International Conference Proceeding Series, Vol. 94. ACM Press, New York (2005) 29–30.
7. Cockton, G., Lavery, D., *A Framework for Usability Problem Extraction*. In Proc. of 7th IFIP TC 13 Int. Conf. on Human-Computer Interaction INTERACT'99 (Edinburgh, August 30-September 3, 1999). IOS Press, Amsterdam (1999) 347–355.
8. Coyette, A., Vanderdonckt, J.: *A Sketching Tool for Designing Anyuser, Anyplatform, Anywhere User Interfaces*. In: Proc. of 10th IFIP TC 13 Int. Conf. on Human-Computer Interaction INTERACT'2005 (Rome, September 12-16, 2005). Lecture Notes in Computer Science, Vol. 3585. Springer-Verlag, Berlin (2005) 550–564.
9. Engelberg, D., Seffah, A.: *A Framework for Rapid Mid-Fidelity Prototyping of Web Sites*. In: Proc. of the IFIP 17th World Computer Congress - TC13 Stream on Usability: Gaining a Competitive Edge WC'2002 (Montreal, August 25-29, 2002). Kluwer Academic Press, Dordrecht (2002) 203–215.
10. Landay, J., Myers, B.A.: *Sketching Interfaces: Toward More Human Interface Design*. IEEE Computer 34,3 (March 2001) 56–64.
11. Lewis, J.R.: *IBM Computer Usability Satisfaction Questionnaires: Psychometric Evalution and Instructions for use*. Int. Journal of Human-Computer Interaction 7,1 (1995) 57–78.
12. Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., Lopez, V.: *UsiXML: a Language Supporting Multi-Path Development of User Interfaces*. In: Proc. of 9th IFIP Working Conf. on Engineering for Human-Computer Interaction jointly with 11th Int. Workshop

on Design, Specification, and Verification of Interactive Systems EHCI-DSVIS'2004 (Hamburg, July 11-13, 2004). Lecture Notes in Computer Science, Vol. 3425. Springer-Verlag, Berlin (2005) 200–220.
13. Meyer, J.: *Creating Informal Looking Interfaces*, 2005, accessible at http://www.cybergrain.com/tech/pubs/lines_technote.html.
14. Newman, M.W., Lin, J., Hong, J.I., Landay, J.A.: *DENIM: An Informal Web Site Design Tool Inspired by Observations of Practice*. Human-Comp. Interaction 18 (2003) 259–324.
15. Paternò, F., Volpe, N.: *Natural Modelling of Interactive Applications*. In: Proc. of 12th Int. Workshop on Design, Specification, and Verification of Interactive Systems DSV-IS'2005 (Newcastle upon Tyne, July 13-15, 2005). Lecture Notes in Computer Science, Vol. 3941. Springer-Verlag, Berlin (2005) 66–77
16. Petrie, J.N., Schneider, K.A.: *Mixed-Fidelity Prototyping of User Interfaces*. In: Proc. of 13th Int. Workshop on Design, Specification, and Verification of Interactive Systems DSV-IS'2006 (Dublin, July 26-28, 2006), Lecture Notes in Computer Science, Vol. 4323. Springer-Verlag, Berlin (2007).
17. Plimmer, B.E., Apperley, M.: *Interacting with Sketched Interface Designs: An Evaluation Study*. In: Proc. of ACM Conf. on Human Aspects in Computing Systems CHI'04 (Vienna, April 24-29, 2004). ACM Press, New York (2004) 1337–1340.
18. Rudd, J., Stern, K., and Isensee, S.: *Low vs. high-fidelity prototyping debate*. Interactions 3, 1 (1996) 76–85.
19. Snyder, C.: *Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces*. Series in Interactive Technologies, Morgan Kaufmann (2002).
20. Vanderdonckt, J.: *A MDA-Compliant Environment for Developing User Interfaces of Information Systems*. In: Pastor, O. & Falcão e Cunha, J. (eds.), Proc. of 17th Conf. on Advanced Information Systems Engineering CAiSE'05 (Porto, 13-17 June 2005). Lecture Notes in Computer Science, Vol. 3520. Springer-Verlag, Berlin (2005) 16–31.
21. Van Laerhoven, K., Schmidt, A., Gellersen, H.-W.: *Pin&Play: Networking Objects through Pins*. In: Proc. of 4th Int. Conf. on Ubiquitous Computing UbiComp'2002 (Göteborg, September 29 - October 1, 2002). Lecture Notes in Computer Science, Vol. 2498. Springer-Verlag, Berlin (2002) 219–228.
22. Virzi, R.A., Sokolov, J.L., Karis, D.: *Usability problem identification using both Low- and High-Fidelity Prototypes.* In: Proc. of ACM Conf. on Human Aspects in Computing Systems CHI'96 (Vancouver, April 13-18). ACM Press, New York (1996) 236–243.
23. Walker, M., Takayama, L., Landay, J.: *High-fidelity or low-fidelity, paper or computer medium?* In: Proceedings of the 46th Annual Meeting of the Human Factors and Ergonomics Society HFES'2002 (Baltimore, September 30-October 4, 2002). Human Factors and Ergonomics Society, Santa Monica (2002) 661–665.