UISKEI: A Sketch-based Prototyping Tool for Defining and Evaluating User Interface Behavior

Vinícius C. V. B. Segura
Departamento de Informática
PUC-Rio
Rio de Janeiro, RJ, Brazil
+55 21 3527-1500 ext 4353
vseguara@inf.puc-rio.br

Simone D. J. Barbosa
Departamento de Informática
PUC-Rio
Rio de Janeiro, RJ, Brazil
+55 21 3527-1500 ext 4353
simone@inf.puc-rio.br

Fabiana Pedreira Simões
Departamento de Informática
PUC-Rio
Rio de Janeiro, RJ, Brazil
+55 21 3527-1500 ext 4353
fsimoes@inf.puc-rio.br

ABSTRACT

Sketching is viewed as an efficient way to design the user interface. However, there are few tools that help designers go from sketching the user interface to simulating its behavior with end-users during early evaluation. We have developed a pen-based tool called UISKEI, which goes beyond allowing designers to define simple navigation between user interface snapshots, into allowing them to define more complex interactive behavior with conditional changes in the state of the user interface. We have conducted two studies on how UISKEI compares to similar prototyping techniques: paper and pencil; widget-based (Balsamiq) and other sketch-based user interface prototyping tools (DENIM and SketchiXML). The studies revealed that UISKEI’s sketch-based definition of the user interface behavior for later simulation is better than the analogous mechanisms provided by the other tools.

Categories and Subject Descriptors
H5.2 User Interfaces: Prototyping; H5.2 User Interfaces: Input devices and strategies;

General Terms
Design

Author Keywords
User interface sketching; early prototyping and evaluation; sketching user interface behavior; pen-based interaction

1 INTRODUCTION

During early user interface (UI) design, we should explore and refine iteratively different design solutions. Sketching on paper is acknowledged as a quick way to document ideas and design details before they are forgotten. However, paper sketches are hard to be kept and modified, particularly in a rapidly evolving scenario, which often results in having to redraw the mockups several times [10].

Incorporating into a computational solution the natural [10], unconstrained [18] and informal [7] virtues of sketching, for which it is praised in many areas [9], could make early prototyping easier and faster. As stated by Gross, “certainly there are many practical benefits to addressing and resolving the challenges of sketch-based interaction and modeling: the design, graphics, and media industries depend heavily on drawing, and being able to engage with artists and designers in their (still) preferred medium of choice is a tremendous advantage” [4].

This paper presents and evaluates UISKEI, a tool that explores how pen-input can be used in a “paperless” UI prototyping application. Analogous to other sketch-based UI prototyping tools, UISKEI maintains the natural characteristic of drawing shapes on paper, and adds the computational power of moving shapes, resizing them and interpreting them as interactive widgets. In addition, UISKEI also features the definition of the prototype behavior, going beyond navigation between UI containers (e.g., windows, web pages, screen shots) into allowing changes to the state of UI elements and widgets (enabling/disabling widgets, for example).

We have conducted two studies on how UISKEI compares to similar prototyping techniques and tools. In the first study, we compared UISKEI with paper-and-pencil and Balsamiq®, a widget-based UI prototyping tool. In the second study, we compared UISKEI with DENIM [13] and SketchiXML [1], two sketch-based UI prototyping tools. The studies revealed that, while UISKEI was equivalent to the other tools with respect to drawing the UI, its sketch-based mechanism for defining the UI behavior for later simulation was significantly better than the analogous mechanisms provided by the other tools.

2 SKETCH-BASED UI PROTOTYPING TOOLS

DENIM is a tool to aid initial states of website design [13]. It features a gesture system that allows some operations to execute directly from the canvas, without switching from the drawing paradigm. Although heavily based on drawing, DENIM still relies on specific tools that work as “stamps” to add WIMP interactive elements (widgets) to the UI prototype. In terms of defining the prototype’s interactive behavior, it is possible to define conditional statements depending on the state of a single widget, but the only available actions are navigational, through hyperlinks. In addition, the conditional statements defined for a page do not reveal which widget is related to each statement.

SketchiXML is a tool focused on UI sketching and its corresponding specification in UsiXML [1]. It has its own gestural language to add elements through drawing. SketchiXML does not support interactive behavior simulation, but allows planning it in a “Navigate” mode. In this mode, the designed UI containers (e.g., pages, windows, and the like) are presented as thumbnails in a 2-D...
space and can be organized and linked to each other through pen input.

Bearing in mind the characteristics of these tools, we have developed UISKEI, a tool that allows both drawing and defining the interactive behavior of a UI through pen-based input. We aimed to lift the restrictions of defining a single condition for an action, and of defining only navigation actions, thus allowing prototyping for more complex UI behaviors. UISKEI also aims at incorporating the pen-based paradigm not only in terms of interface drawing but also in terms of defining the interactive behavior, acting as a front-end tool to reduce learning threshold needed to create interactive prototypes.

3 UISKEI

To build a good UI, we must iteratively refine the solutions, testing it with users to gather feedback and revising them as many times as necessary or possible [16]. This typical interaction design life cycle encompasses a series of inter-related activities in a highly iterative process.

UISKEI (User Interface Sketching and Evaluation Instrument) was developed to aid this life cycle in the early prototyping phase, considering that it can be summarized in three major iterative stages: (i) Interface building; (ii) Behavior definition; and (iii) Prototype evaluation (with users).

During the early prototyping phase, sketching can be highly beneficial due to its inherent speed — allowing rapid exploration and iteration of different ideas — and ambiguity — allowing the designer to focus on basic structural issues rather than format-related details and also allowing multiple interpretations, which can lead to new ideas [13]. Also, the freeform nature of sketches allows the design to be more creative and exploratory than when using the computer [6].

The UISKEI's interface can be seen in Figure 1. It is composed of 4 main areas: the toolbar, the canvas, the filmstrip and the sidebar.

The toolbar allows basic project operations (new, open, save), toggling the UISKEI mode of operation (between drawing without recognition, drawing with recognition, defining behavior and simulation), some tools (such as delete, clone, group and ungroup) and to configure some project default parameters.

The main area of the dialog is occupied by the canvas, where the user can draw the prototype. A red rectangle delimits the size of the presentation unit and the white area stretches in order to contain all drawn elements. Below the canvas there is a filmstrip, containing thumbnails of the presentation units and allowing the user to navigate between them.

UISKEI was written in C# and XAML, using Visual Studio 2010 running in a Tablet PC with Windows 7. The recognition algorithm combines the Levenshtein Distance [5] and the Douglas-Peucker [2] algorithms, defining shapes as strings in an extensible way. The widgets are currently hard-coded but it is planned to support other types of widgets, as well as allowing its customization.

![UISKEI's Interface](image-url)

Figure 1. UISKEI's Interface.
3.1 Interface Building

A prototype in UISKEI is composed of presentation units, which are UI containers (e.g. windows or webpages), which in turn contain UI elements: widgets (recognized elements) or scribbles (unrecognized elements). UISKEI currently supports the following elements: button; checkbox; dropdown; frame; label; radio; spinbox and textbox. It recognizes basic shapes (rectangle, circle, horizontal line, horizontal zig-zag, triangle, vertical line and vertical zig-zag) and may apply some restrictions to them, such as a limit in height and/or width and whether it was drawn inside a specific element (Figure 2). The last restriction is responsible for the “evolution” of elements, a characteristic unique to UISKEI among the researched tools.

Each element has a name, a label, a position, a size, a visibility state, an activation state (enabled/disabled) and additional states depending on the type of widget (e.g. checked, unchecked or mixed for a checkbox). Also, the widget’s initial state is defined to be one shown to the user at the beginning of the simulation.

An event is the action which may trigger an ECA. Each element has its own pre-defined events. While buttons, checkboxes and radio buttons can handle the “Clicked” event, textboxes can handle the “Text Changed” event and the dropdown lists can handle the “Selection Changed” event. For now, an element can only handle one type of event, but future work may overcome this limitation.

Conditions are expressions that must be satisfied in order to trigger an ECA. The conditions currently supported by UISKEI are the following: (a) If the element is in state <state name>; (b) If the element is <visible/visible>; (c) If the element is <enabled/disabled>. An ECA’s set of conditions has an internal operator of “AND”, so all conditions must be met to activate the ECA. The “OR” operator can be achieved by defining a new ECA.

Ideally, the ECAs of an element should be mutually exclusive, since if two or more of them can be activated in a given situation, only the first one in the list will do so. Therefore, the ECAs’ order in ECAMan is important to the simulation.

An action is an operation that will be performed if the ECA is activated, changing the simulation state. While conditions may only refer to elements, an action can refer to an element, to a presentation unit or to a default message. The operations available to each group can be seen in the following list:

- Element actions: (a) Change element to state <state name>; (b) Make the element <visible/visible>; (c) Toggle the element’s visibility status; (d) Make the element <enabled/disabled>; (e) Toggle the element’s enabled status.
- Presentation unit actions: <Change to / Pop up / Modal pop up> the presentation unit
- Default message actions: Show a <information / warning / error> message with <text>

The ECA is represented with a mind-map-like approach. Reflecting the “if / when / do” of an ECA, the “mind-map” shows a block of connections on the left to represent the conditions and a block of connections on the right for the actions. In the middle, the “when” block displays the active ECA and its associated event. By doing so, it is possible to see all the ECA’s conditions and actions in an ordered and centered manner, making it easier to view and edit an ECA, as shown above the canvas in Figure 1.

To add an ECA, the user must be in ECA Mode. While in this mode, the user may select the event triggering UI element and add an ECA to it. The manipulation of an ECA (adding, removing, duplicating, activating the next or the previous ECA) is done by clicking buttons that appear on the canvas when an element is focused, as shown above the canvas in Figure 1.

Conditions and actions can be added to the active ECA by pen gestures, in which the start and end points determine what is being added, according to the following table:

<table>
<thead>
<tr>
<th>Start point</th>
<th>End Point</th>
<th>Creates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>Anywhere</td>
<td>Element condition</td>
</tr>
<tr>
<td>Canvas</td>
<td>Element</td>
<td>Element action</td>
</tr>
<tr>
<td>Presentation Unit</td>
<td>Presentation unit action</td>
<td></td>
</tr>
<tr>
<td>None of the above</td>
<td>Message dialog action</td>
<td></td>
</tr>
</tbody>
</table>

A pie menu allows to define additional parameters for the condition or action being defined.

3.3 Prototype Evaluation

The steps described so far relate to the designer’s perspective, the one who creates the prototype. However, the sketched mockup should also be evaluated according to the end user’s perspective. In order to do so, UISKEI provides a simulation mode, which presents a functional version of the prototype that the user can interact with, in a similar fashion to the final product. The prototype keeps its unfinished look-and-feel, since studies show that testing with rough prototypes does not interfere with the discovery of usability problems when compared to more finished ones [13]. This early stage prototyping can help designers obtain feedback valuable for all design stages [8].

3.4 Supporting Communication in Design

In real world projects, we have found that a major challenge designers face is the need to communicate their design to other members of the development team. To address this challenge, we have incorporated an annotation feature that designers can use to document part of their design rationale and pending issues. Also, UISKEI projects can be exported to an XML file. We are currently...
developing tools to transform a UISKEI markup code into the UI code in other platforms and formats, such as HTML, XAML, and so on, thus making it possible for the designer to communicate with programmers in their “language of choice.”

4 USER EVALUATION STUDIES
The goal of the studies was to evaluate the mechanisms for drawing a UI and for defining its interactive behavior of a simple application fragment: a login process. The test script aimed to replicate iterations of early UI design with several redesigns, albeit in a controlled manner. We asked every participant to perform the same task, thus providing more homogenous results.

Participants were asked to create and simulate a UI prototype, going through three iteration cycles, to build and discuss the effort necessary to build a UI that has one, two, and three checkboxes, with two, four and eight possible target UIs, respectively. The first two cycles were viewed mostly as an opportunity for participants to get to know the tools, and the last cycle was the one that interested us the most, due to its inherent complexity required for defining the prototype’s interactive behavior.

The hypothesis of the studies was that UISKEI should have a poor performance at first, since its language to add elements and ECAs was unknown to most of the participants, but then it would improve in later cycles, as participants learned the language and benefited from having fewer mock-ups with coded behavior.

We conducted two related studies: the first study compared UISKEI to paper prototyping and Balsamiq®, and the second study compared UISKEI to DENIM [13] and SketchiXML [1], all sketch-based prototyping tools. As SketchiXML does not generate a functional version of the prototype, we emulated the paper prototyping technique [17] to simulate the prototype’s interactive behavior in the user evaluation study we conducted.

4.1 Evaluation procedure
The experiments followed a within-group design, comparing the performances of the same participants on all three tools running on a Tablet PC with Windows 7, thus requiring a smaller sample than if each participant was only exposed to a single tool. To avoid the learning effect, we randomized the order in which each participant used the tools, so the learning effect of a user was offset by another. Consequently, the entire data set was not significantly biased by the learning effect [12].

Before using each tool, each participant watched a video introducing the tool and explaining how to add elements and define the interactive behavior in it. In the case of UISKEI, a “cheat sheet” with the gestural language used in defining the UI elements and their behavior (containing Figure 2 and Table 1) was also provided.

After using each tool in each cycle, as well as after the discussion following the 3rd cycle, each participant was asked to answer a short questionnaire, containing ten 5-point scale questions on the following characteristics:

1. Adding UI elements – easy to understand
2. Adding UI elements – effort
3. Difference between resulting UI and specification
4. Adding UI elements – general opinion
5. Defining behavior – easy to plan
6. Defining behavior – efficiency
7. Defining behavior – easy to create
8. Defining behavior – effort to create
9. Defining behavior – general opinion
10. Defining behavior – representation in the UI

All questions were formulated in such a way that higher scores meant better results.

4.2 First Study: Comparing UISKEI to paper prototyping and Balsamiq®
The first study compared UISKEI to other two prototyping techniques: paper prototyping [17] and widget-based prototyping using Balsamiq®. This study involved 12 young participants: seven with a background in Computer Science, one in Economy, one in Industrial Engineering, two in Electrical Engineering, and one in Design. Only one of them had never designed a UI before, and five of them had to face this task several times per month. We expected that Balsamiq® would perform well in the beginning, due to its extensive collection of widgets and the well-known drag-and-drop paradigm, but the need to duplicate screens to define and show the behavior by using only navigation would make it harder to use as complexity increases.

After performing the tasks, participants went through a quick interview, asking them about which tool they would use in the following situations, and why:

11. In an early development stage, while exploring the idea space, where different solutions are considered and constantly changed, focusing only in the interface.
12. When the idea is clearer, a solution was chosen and a single prototype needs to be built, focusing still only in the interface.
13. Considering that they needed to define the behavior of the chosen solution.

4.2.1 Evaluation results
Overall, the hypothesis of an increase in the scores given to UISKEI in later cycles was confirmed by the questionnaire results, shown in Figure 3. After the end of the second cycle, UISKEI only received scores lower than the other tools in Q5 and Q7, showing that the logic behind defining ECAs is not easily grasped at first. By the end of the third cycle, however, UISKEI’s average scores in all questions were greater than the other tools and all greater than 4.0 (within the scale of 1 to 5).
The first column indicates the questions; the following three columns indicate the score average for each tool used in the study; the fourth column indicates the ANOVA’s F value; the fifth, the critical interval - calculated using the Tukey-Kramer procedure with a 95% confidence level -; and the last two columns indicate the average differences when comparing UISKEI to Balsamiq (U-B) and UISKEI to paper prototyping (U-P), respectively. The relevant values are highlighted in boldface.

**Table 2. ANOVA of the 3rd cycle questionnaire results.**

<table>
<thead>
<tr>
<th>Question</th>
<th>Paper</th>
<th>Balsamiq</th>
<th>UISKEI</th>
<th>F</th>
<th>p</th>
<th>Critical Interval</th>
<th>U-B</th>
<th>U-P</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>4.83</td>
<td>4.67</td>
<td>4.83</td>
<td>0.306</td>
<td>*</td>
<td>1.127</td>
<td>1.250</td>
<td>1.250</td>
</tr>
<tr>
<td>02</td>
<td>3.42</td>
<td>3.42</td>
<td>4.67</td>
<td>4.853</td>
<td>*</td>
<td>11.27</td>
<td>1.250</td>
<td>1.250</td>
</tr>
<tr>
<td>03</td>
<td>3.58</td>
<td>3.92</td>
<td>4.75</td>
<td>2.746</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>04</td>
<td>2.83</td>
<td>3.50</td>
<td>4.67</td>
<td>10.230</td>
<td>**</td>
<td>0.998</td>
<td>1.167</td>
<td>1.833</td>
</tr>
<tr>
<td>05</td>
<td>4.25</td>
<td>4.67</td>
<td>4.50</td>
<td>0.443</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>06</td>
<td>2.83</td>
<td>2.08</td>
<td>4.42</td>
<td>10.125</td>
<td>**</td>
<td>1.288</td>
<td>2.333</td>
<td>1.583</td>
</tr>
<tr>
<td>07</td>
<td>3.83</td>
<td>3.58</td>
<td>4.17</td>
<td>0.603</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>08</td>
<td>3.33</td>
<td>2.75</td>
<td>4.17</td>
<td>3.678</td>
<td>*</td>
<td>1.277</td>
<td>1.477</td>
<td>0.833</td>
</tr>
<tr>
<td>09</td>
<td>2.17</td>
<td>1.67</td>
<td>4.42</td>
<td>27.192</td>
<td>**</td>
<td>0.966</td>
<td>2.750</td>
<td>2.250</td>
</tr>
<tr>
<td>10</td>
<td>2.67</td>
<td>2.42</td>
<td>4.58</td>
<td>8.367</td>
<td>**</td>
<td>1.410</td>
<td>2.167</td>
<td>1.517</td>
</tr>
</tbody>
</table>

The lowest scores were given in Q7 and Q8, suggesting that participants faced difficulties in handling new behaviors with all the tools. However the answers to Q8 reveal that the participants considered UISKEI more effortless to solve these problems. The ANOVA reinforces this assumption when comparing UISKEI to Balsamiq, since the difference between their averages (U-B) was higher than the critical interval.

In the same cycle, the questions with the biggest differences from other tools were related to the interaction definition: Q6, Q9, and Q10, all p<0.01. In general, participants liked having a single interface (contrary to the multiple ones created in Balsamiq) and a previously defined behavior (opposed to the “on-the-fly” simulation of paper). Q10 results also show that the mind-map representation of ECAs was well accepted.

Another good indicator of UISKEI’s success was the answers to the “hated it / loved it” questions Q4 and Q9, both of them with p<0.01. Q4 is related to the UI and shows that the added complexity is quickly perceived in the paper technique, which faces an almost steady decrease in its scores in all cycles, while Balsamiq decreases only in the last cycle (see Figure 3). UISKEI on the other hand, has a steady increase of its scores, showing that the language to add elements is well appreciated by users. The “hated it / loved it” interaction question (Q9) showed yet another pattern, with a steep decrease in the last cycle for both paper and Balsamiq, while UISKEI received an almost constant score, showing that users liked the way that the increased simulation complexity was handled.

The ANOVA results show that, from the ten questions asked, we obtained significant results in six of them, in which the average difference observed between UISKEI and the other tools is always positive (so UISKEI’s scores were greater than the other tool’s scores) and is significant in most cases. The only exception is the U-P difference in Q8, showing that the effort in defining behavior is not significantly different between UISKEI and paper.

The interview results were also in favor of UISKEI, as can be seen in Table 3, where the numbers in parenthesis show how many people chose the tool for each situation. In I1, while 25% of participants chose paper, 33% of them chose UISKEI. Balsamiq’s results in I1 and I2 may be a result of its vast library of elements and features, such as alignment options and gridlines. However, the power of ECAs is shown in the answers to the I3, in which the vast majority chose UISKEI over the other two tools.

![Figure 3. Average grades (Y-axis) by cycle (X-axis).](image-url)
Table 3. Interview results.

<table>
<thead>
<tr>
<th></th>
<th>I1</th>
<th>I2</th>
<th>I3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Balsamiq Paper</td>
<td>42% (5)</td>
<td>83% (10)</td>
<td>0% (0)</td>
</tr>
<tr>
<td>UISKEI</td>
<td>25% (3)</td>
<td>0% (0)</td>
<td>8% (1)</td>
</tr>
<tr>
<td></td>
<td>33% (4)</td>
<td>17% (2)</td>
<td>92% (11)</td>
</tr>
</tbody>
</table>

4.2.2 Participants' opinions

Most participants were not familiar with paper prototyping. After seeing the introductory video, some participants already complained about the work involved. This kind of prejudice may explain the corresponding low scores. When they assumed the “computer” role during the prototype simulation, the number of complaints increased. P6 stated that “using paper may cause confusion in the ‘computer’”. This opinion was later reinforced by P8, who called him/herself as a “486” (referring Intel’s older line of microprocessors) during paper simulation, summing up that “Paper is fun, but not much practical”. The overall opinion about drawing on paper was that it is good for a rough sketch, but difficult to make changes. It can be summarized in P1’s declaration: “it really complicates in the sense that you don’t have too much flexibility once you have already drawn something. I think that you become too restricted to what you have or you start from scratch, which is certainly not efficient”. This shows a disadvantage of paper, since the iterative refinement of different solutions may be discouraged due to the effort of making changes in the prototype. Regarding the paper simulation, some terms used were “boring”, “disgusting” and “hell.” P4 pointed out that: “despite being easy to do and easy to simulate, you don’t have a record of what was happening, unless you film it, of course, but even so, you don’t have the register of the used logic or even the errors that happened.”

Balsamiq, on the other hand, divided opinions, ranging from people who loved it and others who hated it. Amongst the former, the most praised features were the smart gridlines and the overall look-and-feel of the elements, showing that once you have these “aesthetics” facilities, they turn into a major concern. Actually, comparing the Balsamiq prototypes with the other ones, the disposition of the elements was much more similar to the ones depicted in the test script. This focus on the detailed look-and-feel was not observed in UISKEI, since, as a sketching tool, the focus should be more in the overall interaction and structure rather than on its “aesthetics” [11]. The latter group focused their dislikes on the limitation of navigational actions, having to duplicate the prototype due to the lack of conditionals. Participant P11 said that “it is hell having to replicate (...) if there isn’t an ‘if’, nothing works”. In comparison with UISKEI, it was considered “less dynamic”, as said by P5, who simply said “Balsamiq is static, UISKEI is dynamic”.

Regarding UISKEI, the most common comment was about the “learning curve” and the terminology used. This visual programming problem was already stated in [15]: “the tools for producing these applications often require months or even years of study to use effectively, and more often than not require the use of programming languages that are difficult to use even for professional programmers”. However, the participants envisioned that, once learned, it would make the design process easier. P11 said that “in UISKEI, the learning curve is a little high, a little higher than usual, but once you get it, it is piece of cake”. P4 added “it is something that you can take a while to learn, but once you learn it, it will be way faster to use”.

The behavior definition process fits the “learning curve” observation, since most users are not familiar with either pen-based interaction or pie menus. Besides the difficulties, it was well accepted, as described by P8: “I found UISKEI quite cool. I liked this way of connecting events, of creating conditions and actions, way practical and well integrated to the pen, easy to use even without mouse and even with the pen not being so precise”. This participant was not the only one who complained about pen interaction issues: P1 missed the use of keyboard, since writing recognition was not featured in Brazilian Portuguese (language of the test). The ECA representation was also praised. Comparing to Balsamiq and its replicated interface, P4 said that “UISKEI’s way is more interesting, because you can see all the conditions that are happening at the same time”. P2 praised the way the ECAs were presented, saying that “what I liked is that it is very compact, you can work in an organized way (...). The interface is compact, things are shown in the right place and the actions are simple”.

4.3 Second Study: Comparing UISKEI to DENIM and SketchiXML

A second study, with 10 Computer Science students or professionals, compared UISKEI to other two sketch-based prototyping tools: DENIM [13] and SketchiXML [1]. Once again, the goal was to evaluate both the UI design and the behavior definition, so we have used the same test script used in the first test, with minor changes: participants were now asked to perform the task in only two cycles of iteration, the 1st cycle with one checkbox and the 2nd cycle with three checkboxes.

Before using each tool in the first cycle, once again videos were shown to introduce the tools and their mechanisms to add interface elements and to define the prototype’s interactive behavior. In addition to the “cheat sheet” with UISKEI’s language provided in the first study, two other “cheat sheets” were provided, presenting the languages used in SketchiXML and in DENIM. After using each tool in both cycles, participants were asked to answer the same questionnaire used in the first test. No interview was conducted after participants finished the tasks, but some spontaneous opinions were collected during the session.

4.3.1 Evaluation Results

A single-factor ANOVA for the 2nd cycle results is presented in Table 4, in which the critical interval was calculated using Tukey-Kramer with 95% confidence level.

Table 4. ANOVA of the 2nd cycle questionnaire results.

<table>
<thead>
<tr>
<th>Question</th>
<th>DENIM</th>
<th>SketchiXML</th>
<th>UISKEI</th>
<th>F-value</th>
<th>p-value</th>
<th>Critical Interval</th>
<th>E-D</th>
<th>E-S</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>4.40</td>
<td>4.80</td>
<td>3.90</td>
<td>1.766</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>3.70</td>
<td>3.30</td>
<td>4.30</td>
<td>1.994</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>03</td>
<td>2.90</td>
<td>3.20</td>
<td>4.40</td>
<td>5.505</td>
<td>**</td>
<td>1.181</td>
<td>1.50</td>
<td>2.00</td>
</tr>
<tr>
<td>04</td>
<td>3.50</td>
<td>3.10</td>
<td>4.30</td>
<td>4.289</td>
<td>**</td>
<td>1.030</td>
<td>0.80</td>
<td>1.20</td>
</tr>
<tr>
<td>05</td>
<td>4.10</td>
<td>3.40</td>
<td>3.60</td>
<td>0.983</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>06</td>
<td>2.70</td>
<td>1.60</td>
<td>3.10</td>
<td>6.962</td>
<td>**</td>
<td>1.027</td>
<td>0.40</td>
<td>1.50</td>
</tr>
<tr>
<td>07</td>
<td>3.00</td>
<td>1.90</td>
<td>3.10</td>
<td>3.344</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>08</td>
<td>3.90</td>
<td>1.90</td>
<td>4.10</td>
<td>23.928</td>
<td>**</td>
<td>0.868</td>
<td>0.20</td>
<td>2.20</td>
</tr>
<tr>
<td>09</td>
<td>3.50</td>
<td>1.70</td>
<td>3.80</td>
<td>15.779</td>
<td>**</td>
<td>1.030</td>
<td>0.40</td>
<td>2.20</td>
</tr>
<tr>
<td>10</td>
<td>4.10</td>
<td>2.30</td>
<td>4.30</td>
<td>14.182</td>
<td>**</td>
<td>1.021</td>
<td>0.20</td>
<td>2.00</td>
</tr>
</tbody>
</table>

UISKEI’s lowest average scores were given in Q6 and Q7, both with an average score of 3.10, followed by Q5, with an average score of 3.60. This shows once again that the logic behind the interactive behavior definition process is not easily understood at first. Nevertheless, when compared to the scores given to the other tools in the same questions, UISKEI’s scores were still somewhat higher. This suggests that participants faced difficulties when
dealing with behavior in all the tested tools, even though the answers to Q8 show that UISKEI was considered significantly more effortless by the participants. This analysis reinforces the fact that participants preferred to define the behavior with UISKEI. Comparing it to SketchiXML, in all questions related to behavior definition, the average differences between UISKEI and SketchiXML (U-S) were greater than the calculated critical interval.

Compared to the other tools, UISKEI was given a lower average score only in Q1 and Q5, showing that participants had difficulties understanding the language used to define widgets and behavior. Still, ANOVA cannot disprove the null hypothesis, meaning that the tools are not significantly different in this respect. Also, the answers to Q3 show that the prototypes created with UISKEI were more similar to what participants had in mind than the prototypes created with the other tools, significantly more similar than the prototypes created with SketchiXML. In the answers to Q4, related to how participants liked the interface drawing process, UISKEI was given an average score of 4.30, significantly better than SketchiXML (3.10), but not significantly different from DENIM (3.50).

From all 10 questions asked to participants, in five of them UISKEI is significantly better than SketchiXML, and in one of them it is significantly better than both. Comparing to DENIM (U-D), UISKEI achieved significant differences only in Q3, related to the correspondence between the intended and obtained UIs. This suggests that using UISKEI is not significantly different from DENIM for most evaluated tasks.

4.3.2 Participants’ Opinions

DENIM, despite its good scores, divided opinions. Participants who enjoyed it listed, amongst its advantages, that it is “fast and easy” to elaborate a prototype —indeed, DENIM cycles were shorter than the ones with SketchiXML and UISKEI. On the other hand, participants who did not like it stated that the prototype created with DENIM looked “messy”. They complained about the “stamps” used to create some widgets, saying that they were not precise enough: “It’s impossible to put things where I want” (P1).

In fact, DENIM had the lowest scores in Q3, related to how similar the prototype was to what participants had in mind. P1, after using DENIM in the 2nd cycle, stated: “I’m awful with pen and paper, I have no coordination. I was hoping this could have coordination (sic!) for me”. Also, the gesture language wasn’t much appreciated by participants, but it is unclear whether that was related directly to the language or to the gesture recognition (some participants were unable to reproduce the gestures, and so kept using the menus). In terms of behavior definition, P3 summarized most participants’ main complaints about DENIM: “I can’t use the pen to do this. The button to change from one condition to another is impossible to click on”. Participants spent most of their respective DENIM cycles repeatedly trying to click the arrow (what P3 called “button”) and P6 said the process was not “pen-friendly”. Also, there were complaints about the zoom level necessary to draw the arrows to connect widgets from different pages. Most participants wanted to zoom out to do this, but could only do it at the page level. However, overall, participants enjoyed the behavior definition method and were pleased with the representation of the behavior. Participants felt the arrows allow easily understand of connections and “flow”.

As SketchiXML does not generate some sort of interactive prototype, it was necessary to emulate a paper prototyping technique with the screens created to simulate interaction. After watching the introductory video, even before the 1st cycle, participants were feeling optimistic about the technique, since it looked “easy” and “uncomplicated”. But the complaints already started during the 1st cycle, in which participants said the method was “confusing”. Complaints increased in the 2nd cycle, when participants faced a scenario with eight possible outcomes. Most of them stated they were “upset” about re-drawing the screen eight times to represent all possible combinations of the three checkboxes values. In the end of the 2nd cycle, participants were not satisfied with the process and 2 participants even gave up the task before concluding the simulation (P7 and P10), showing that reproducing the paper prototyping experience in a computational environment is not ideal. The overall opinion about drawing the UI using SketchiXML was that it is very limited. Most participants were satisfied with the widget language and the recognition system, but they felt it did not support free drawing and handwriting very well: “I can’t draw a line without it telling me that it is a label. I can’t do simple things. Can’t I turn recognition off?” (P4).

Regarding UISKEI, participants faced difficulties to understand both the UI drawing and behavior definition processes. As in the first test, the most common comment was about its “learning curve” and terminology, especially when it comes to defining the behavior. Indeed, after watching the introductory videos, participants seemed more confident about using both DENIM and SketchiXML than about UISKEI. Also, interactions with UISKEI on the 1st cycle were longer than interactions with the other two tools, because participants had some issues dealing both with the logic behind ECAs and the language for drawing widgets. However, when it came to actually defining the behavior, participants later envisioned that, once learned, UISKEI made the process easier, as stated by P8: “after I got it, it felt just like telling the program what I wanted”.

In terms of UI drawing, the main complaint was about the limits in height and width to draw some widgets. Some participants had a hard time trying to conform their “drawing style” to UISKEI’s constrains for recognition. Apart from that, participants enjoyed manipulating the elements (by cloning, moving, editing) and the overall look of the prototype. P4, P8 and P10 stated that the only prototype they would feel comfortable to show to a colleague or a client was the one created in UISKEI. To justify that, P4 said that “I know it’s not supposed to be final, but there’s a certain level of mess I can deal with and here I can manage it”. In terms of behavior definition, even though participants complained about the learning curve and faced ergonomic issues when interacting with the pen and the pie menu, ECAs were well accepted, especially when it comes to their representation. P8 said that “the representation compensates the hard time I had; it’s really easy to understand what’s going on”. Participants also appreciated the range of actions that could be defined and not being restricted to navigational actions.

5 CONCLUSION

UISKEI was developed to allow the user to draw UI elements using pen input, both maintaining the natural characteristic of drawing shapes on paper, and adding the computational power of moving shapes, resizing them and interpreting them as interface elements. UISKEI also features the pen-based definition of the prototype’s behavior, by means of an event-conditions-actions structure. It allows designers to go beyond simple navigation between UI containers (e.g. windows, web pages, screen shots) and to define changes to be made to the UI elements and widgets (enabling/disabling widgets or changing their state, for example)
when the user interacts with the prototype being designed in the simulation mode.

In the evaluation studies, participants had an overall positive opinion about UISKEI, in particular with respect to paper prototyping, Balsamiq, and SketchiXML. Although UISKEI allows the definition of more complex behavior, its mechanism was not found to require more effort than any of the tools, including DENIM. Moreover, we hypothesize that, for prototyping more complex behaviors, there will be a significant difference between UISKEI and all other tools, because the latter require the creation of many more UI instances for simple variations in the UI states. The studies were quite limited, however, in that the tasks were very simple and focused on building a predefined user interface, as opposed to a more realistic open-ended design activity. Future studies will be conducted to evaluate this aspect, and also to compare UISKEI with commercially available prototyping tools.

The study revealed some opportunities for improvement, mainly to soften UISKEI’s learning curve. In the next version of UISKEI, it will present better support in the form of contextual help to remind users of the different gestures they can use in each mode UISKEI supports (drawing, recognition and ECA); allow customizing some of the constraints used by the recognizing algorithm; and fine tune the interaction with the pie menu. These changes will be evaluated in a longitudinal study.

6 ACKNOWLEDGMENTS
The authors would like to thank CNPq for the financial support to their work.

7 REFERENCES