2
Related work

Several tools can aid the prototyping stage, with many approaches available: desktop or web-based applications, UI-specific or generic diagrammatic solutions, mouse-based or pen-based interaction, amongst others. Lists of some products can be found online, for example, in (Harrelson, 2009) or (Barber, 2009). An overview of some widely used applications will be presented in the following sections. Section 2.1 analyzes mouse-based prototyping software and Section 2.2, pen-based ones. Section 2.3 presents a comparison table summarizing the analyzed applications.

2.1
Mouse-based prototyping software

2.1.1.
Microsoft Visio

Microsoft Visio[1] is a tool that allows the creation of various types of diagrams - from flowcharts to Windows-style user interfaces. For each type, a pre-defined set of elements is presented and, by drag-and-drop, the selected ones are added to the representation being built.

Although largely known and utilized, Microsoft Visio presents relevant limitations. Since it handles a wide array of diagram types, it is a rather generic solution to the UI prototyping problem. Therefore, specific behavior is not supported, allowing only the navigation between screens. Its strength resides in the fact that the mockup interface is really similar to the final result in a Windows XP environment, as seen in Figure 1.

[1] Information about the latest version can be found at http://office.microsoft.com/en-us/visio/

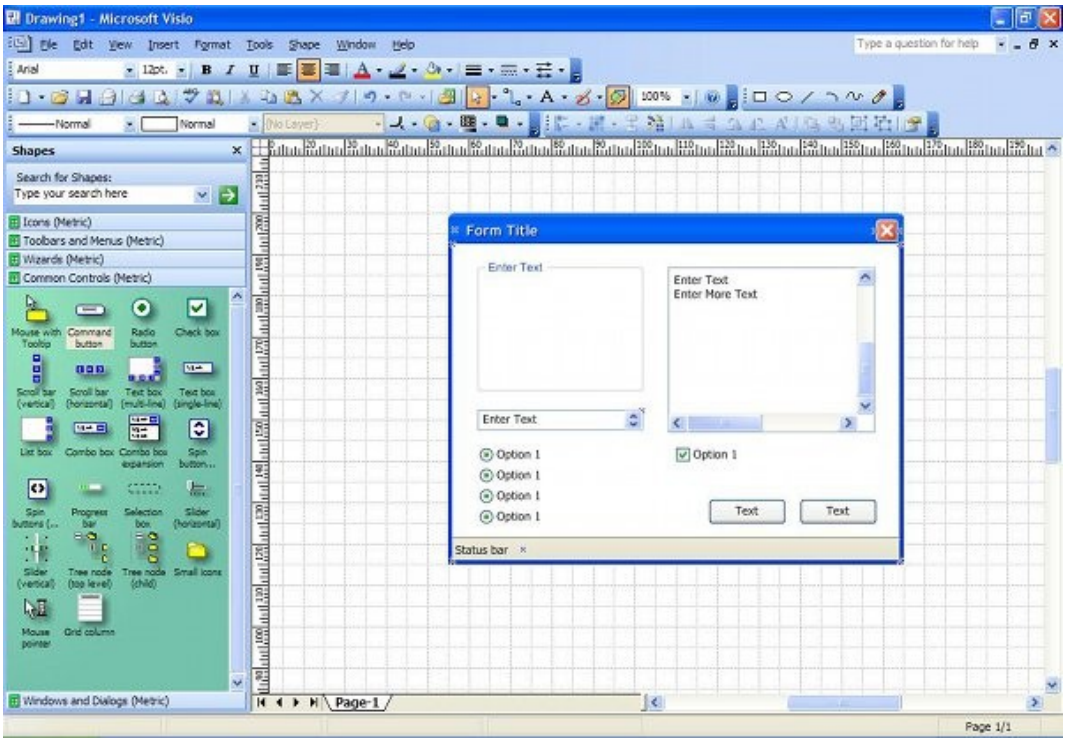

Figure 1: Microsoft Visio 2007.

2.1.2. Balsamiq

Balsamiq[2] is a very popular UI prototyping software as of the writing of this dissertation. It presents a collection of 75 elements that can be added to the mockup, varying the complexity from simple buttons and labels to more complex ones, such as a formatting toolbar or an iTunes-like cover flow.

Elements are added by drag-and-drop and the only possible interaction with them is as hyperlinks between mock-ups (for example, a checkbox cannot toggle its "checked" state during the "full-screen presentation"). This limitation is a major disadvantage of the software, since if the designer wants to create an interactive prototype, he/she must build copies of the same interface with the elements in different states and then create links between these mock-ups. Moreover, not all elements can have hyperlinks associated to them. For instance, cover flow, numeric stepper and playback controls cannot trigger navigation actions.

Although not pen-based, Balsamiq's elements have a "sketchy" look-and-feel, which can be seen in Figure 2. This helps to enhance the conceptual

[2] http://www.balsamiq.com/

difference between “prototype” and “product” to the final user, thus “can help to disarm those who think that suddenly your software is ‘done’” (Harrelson, 2009).

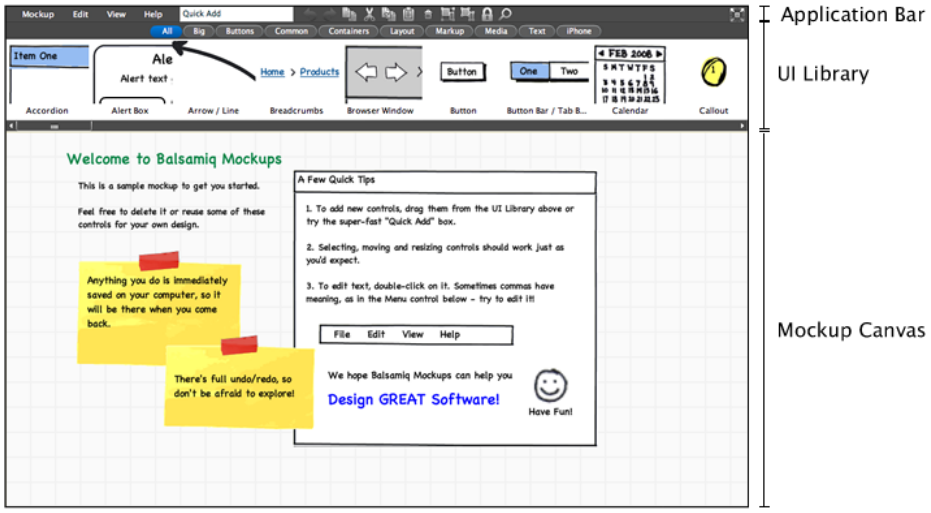

Figure 2: Balsamiq[3].

2.1.3. Axure RP Pro

Another UI prototype tool is the Axure RP Pro[4]. It supports defining other forms of interaction than only navigating between screens, allowing the generation of a functional prototype with less effort. Its interface can be seen in Figure 3.

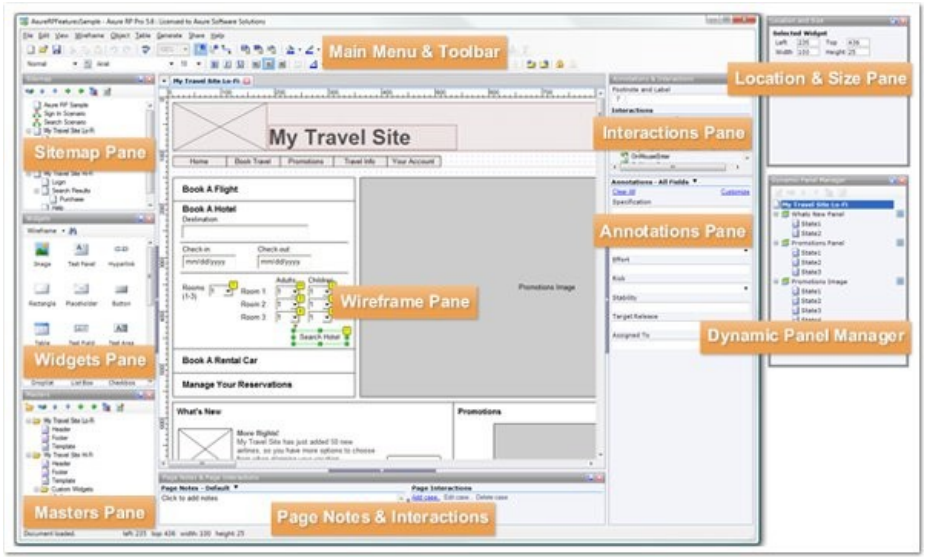

Figure 3: Axure RP Pro[5].

[3] Image taken from: http://balsamiq.wpengine.netdna-cdn.com/images/help_3mainareas.png
[4] http://www.axure.com/
[5] Image taken from: http://www.axure.com/images/training-axurerpenvironment-interface.jpg

The addition of elements is also done by drag-and-drop and there is an extensive list of properties to be defined for each element. The form-based paradigm extends to the definition of the interactive behavior, being necessary to fill out some fields and requiring several mouse clicks, as can be seen in Figure 4.

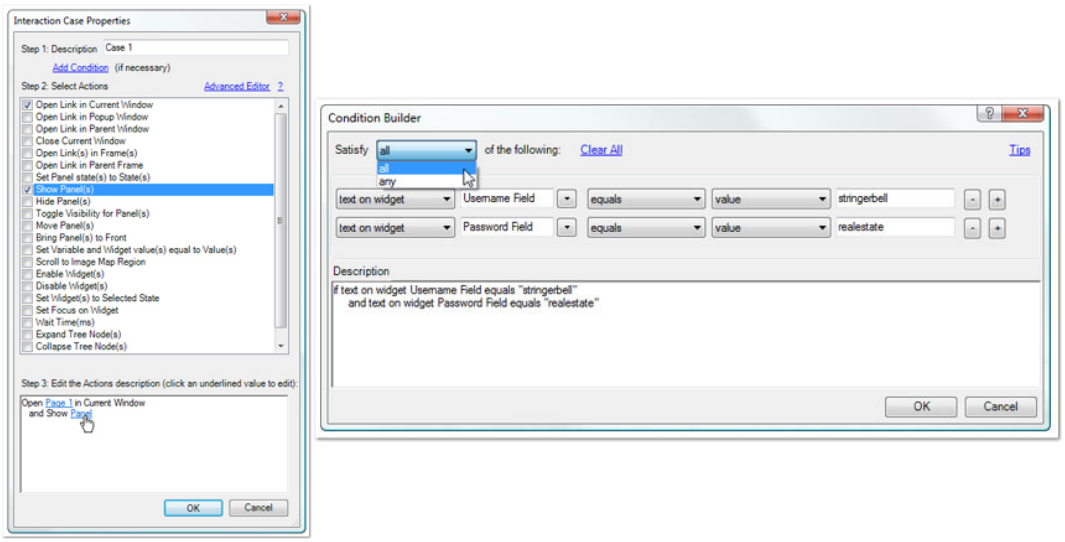

Figure 4: Defining behavior with Axure RP Pro[6].

2.2
Pen-based prototyping software

2.2.1.
DENIM

DENIM [7] (Lin, Thomsen, & Landay, 2002) explores the pen-based interaction paradigm to aid the initial states of website development. Its main characteristic is the different zoom levels to view the project, going from a macro vision – the site map – to a micro vision – a single page. The pen strokes easily create links between pages by dragging lines between them, as can be seen in Figure 5, in which the "Home" page links to the "Weather" page.

[6] Images taken from: http://www.axure.com/images/training-interactions-dialog.jpg and http://www.axure.com/images/training-conditionallogic-multipleconditions.jpg

[7] http://dub.washington.edu:2007/denim/

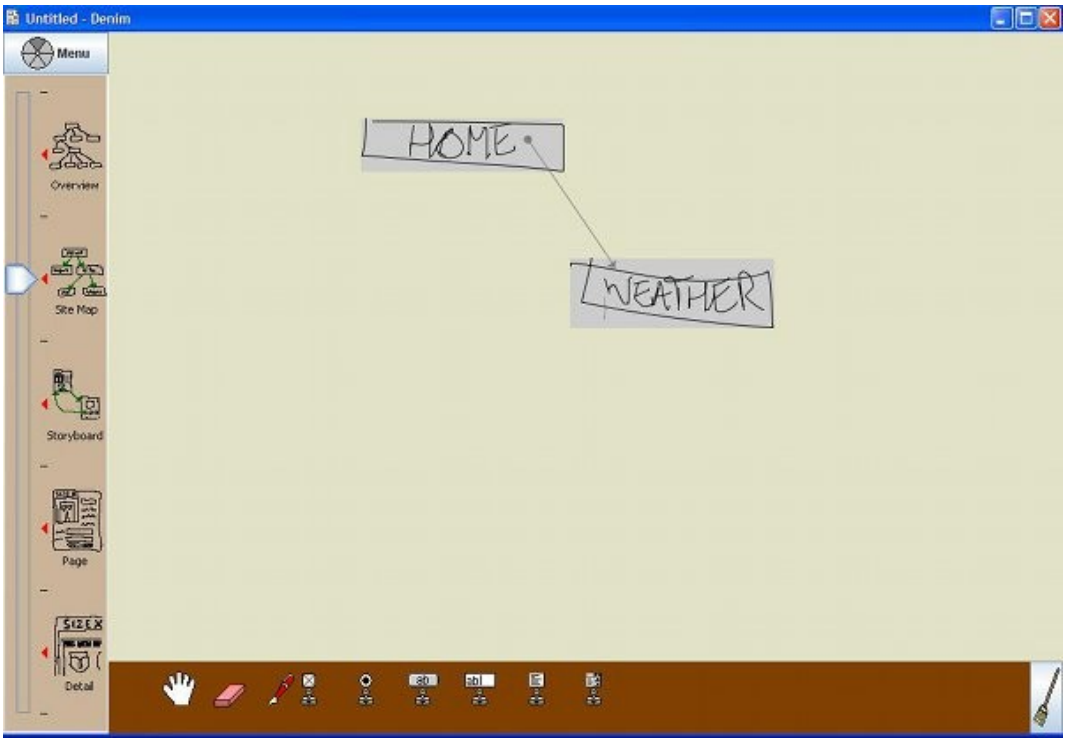

Figure 5: DENIM.

DENIM also features a gesture system that naturally flows along with the drawing of pages. If the user holds the pen's barrel button or the `CTRL` key, the drawing will be interpreted as a gesture, following the language presented in Figure 6. The gesture system allows frequent operations, such as undo/redo and cut/copy/paste, to be executed directly from the canvas, without changing the drawing paradigm by adding an implicit mode of interaction (a "gesture mode" activated by the holding of the pen's barrel button or the `CTRL` key).

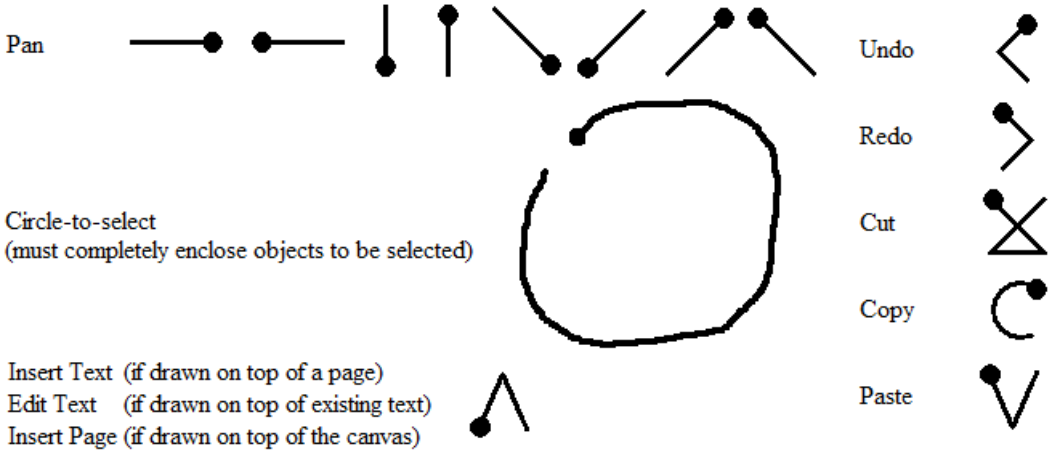

Figure 6: DENIM gesture system[8].

[8] Images taken from online documentation available at : http://dub.washington.edu:2007/projects/denim/docs/HTML/quick_ref/gestures.html

However, even heavily based on drawing, the addition of WIMP (Windows, Icons, Menus and Pointers) elements still relies on specific tools that work as "stamps", as can be seen in the lower bar of Figure 5. Another limitation is that the only available actions are navigational (hyperlinks), but it is possible to make a conditional navigational depending on the state of elements. Such conditionals are displayed one at a time, without highlighting which component is related to each conditional. As can be seen in Figure 7, the checkbox is responsible for the two possible navigational paths, but it is not highlighted in any way.

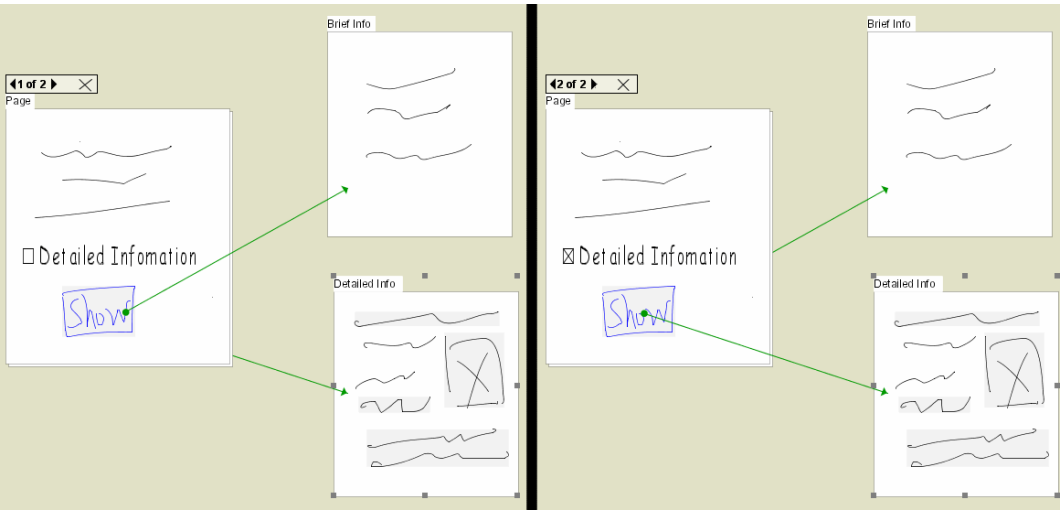

Figure 7: DENIM's representation of conditionals[9].

One nice feature is the idea of "custom component", allowing the use of a user-defined element in the application. The operations regarding custom components — such as creating, adding and editing — are accessible through the pie menu, so it is not possible to add these components through drawing or stamps as the regular ones.

2.2.2. SketchiXML

The SketchiXML[10] is a "multi-agent application able to handle several kinds of hand-drawn sources as input, and to provide the corresponding specification in UsiXML" (Coyette, Faulkner, Kolp, Limbourg, & Vanderdonckt, 2004). It focuses on UI sketching and has its own gestural language to add elements through drawing, which can be seen in Figure 8.

[9] Images taken from online documentation available at: http://dub.washington.edu:2007/projects/denim/docs/HTML/tutorial/Using_Conditional.htm

[10] http://www.usixml.org/index.php?mod=pages&id=14

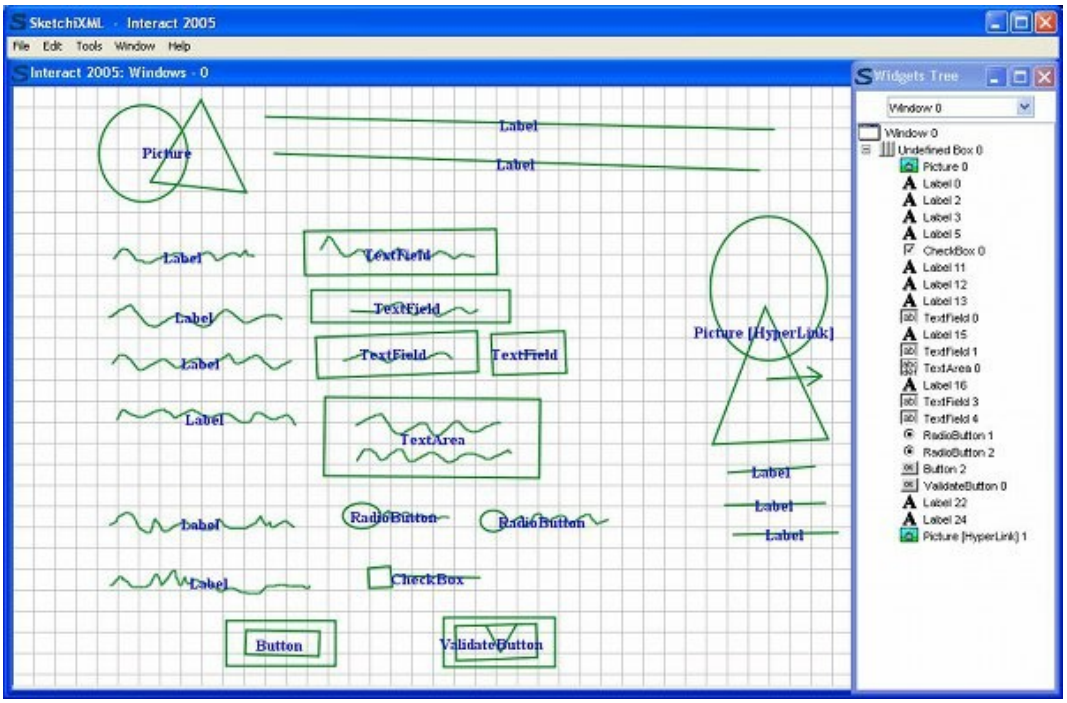

Figure 8: SketchiXML[11].

Not all elements can trigger actions. For instance, a button can trigger multiple actions, but an image can trigger none. Moreover, these actions are limited to navigation between screens. This behavior definition is done in the "Navigate" mode, where the screens are presented as thumbnails in a 2-D space. Then they can be organized by the user, since he/she is unaware of this 2-D space when he is building the screens. The addition of actions explores the pen input, since we need to draw a line connecting the element that will trigger the action to the screen that will be shown. When a valid connection is available, the line being drawn changes its color, giving feedback that an action can then be created. After lifting the pen, a pop-up menu appears so the user can choose what action will be created (open/close, minimize/maximize, bring to front/back). The creation of an action therefore happens in two steps: drawing a line to determine the trigger element and the target screen and then selecting the action from the pop-up menu.

A particular characteristic of SketchiXML is that it has different levels of the mockup visual representation: the original stroke, a "beautified" stroke, a conceptual version of the element and the element as would be shown at the interface of the current running operating system. The different representation levels can be seen in Figure 9, starting from the top-left corner and going clockwise:

[11] Image taken from: http://www.usixml.org/images/sketchixml_09.png

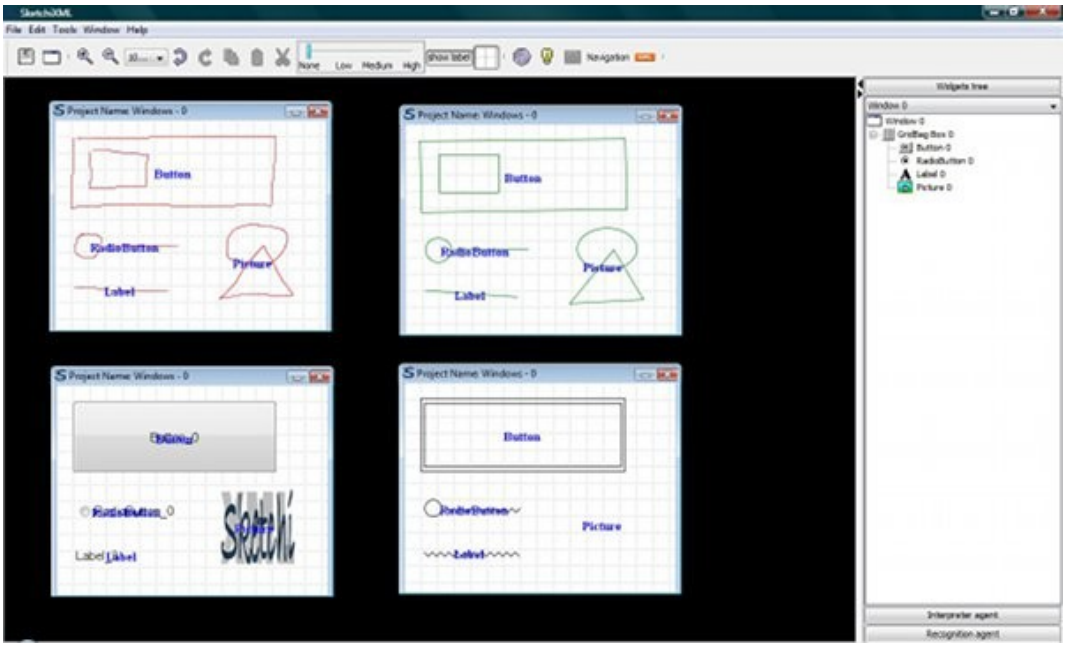

Figure 9: Different levels of representation in SketchiXML.

2.2.3. CogTool

CogTool[12] is a software currently being developed by the Carnegie Mellon University that, besides creating UI prototypes, "automatically evaluates your design with a predictive human performance model" (Carnegie Mellon University, 2009). One difference from all other evaluated tools is that CogTool supports different input devices (not only the usual keyboard and mouse, but also touch screen and microphone) and audio as another output device in addition to the monitor screen.

In CogTool, a project consists of frames, related to the windows of the interface being designed. The elements are defined by drawing a rectangular area that will be occupied by it. Despite having this rectangle drawing component, the user must choose the corresponding tool to choose the element type.

Between frames it is possible to add a transition, having an event (such as a mouse or keyboard input) associated to it. These transitions can be shown in a storyboard-like fashion, allowing an overview of the project and the relations between frames, as can be seen in Figure 10.

[12] http://cogtool.hcii.cs.cmu.edu/

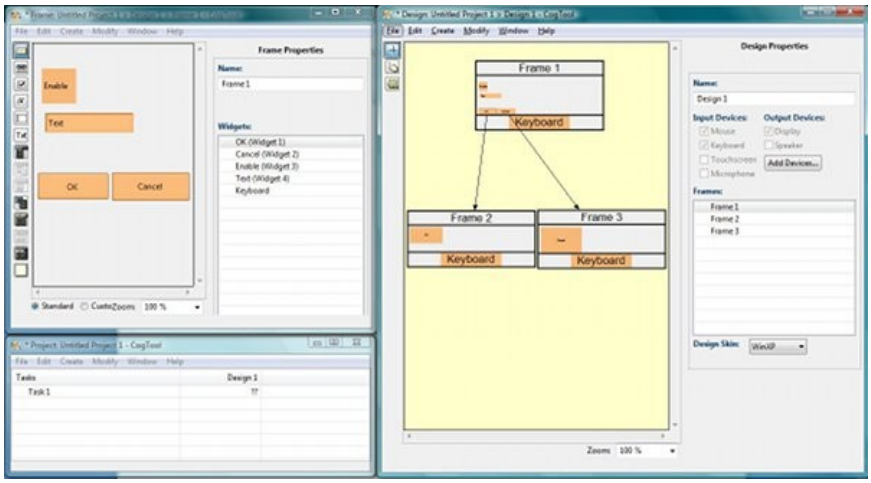

Figure 10: CogTool.

Having defined the frames and transitions, it is possible to make a GOMS task analysis simulation with a "cognitive crash dummy" (as described in the project's webpage), measuring the time elapsed in each step. In the end, a graph summarizing the results is displayed, as shown in Figure 11.

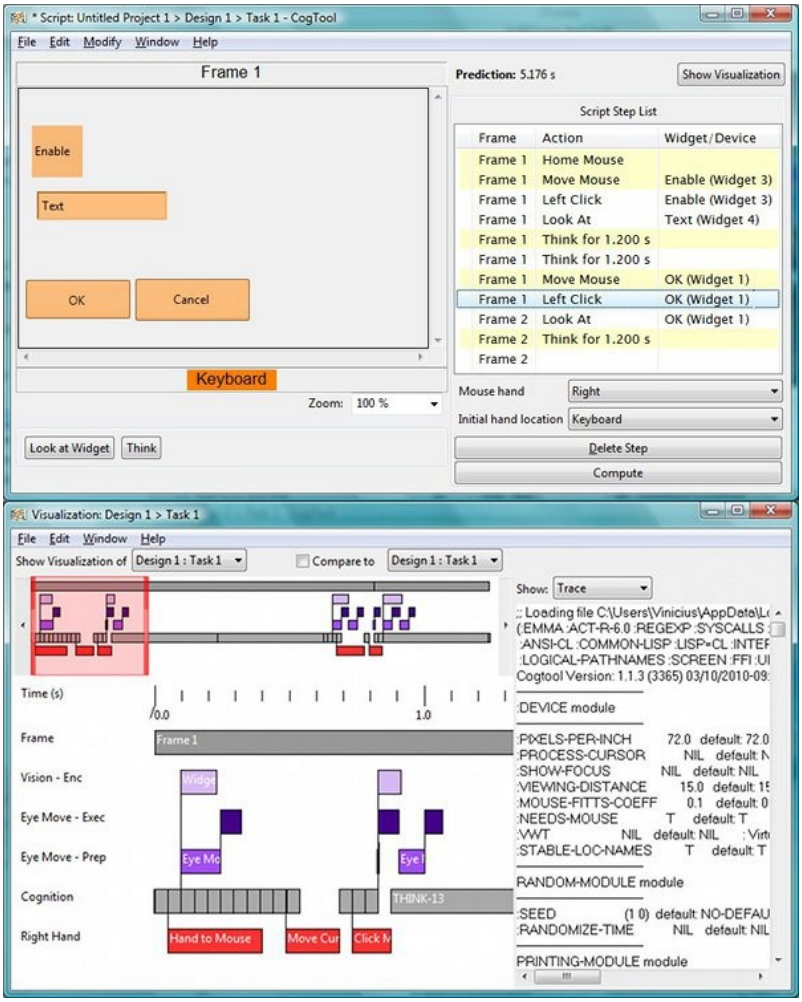

Figure 11: Simulation in CogTool.

Besides a good task analysis, the prototype is very simplified, since all user interface elements are graphically represented by rectangles (bounding boxes) with which users can interact. As the main focus is the automated task analysis to be used by the design team, the choice for this simplified representation is justified. However, if the prototype is presented to an end user, we believe that this design choice of only displaying bounding boxes could result in some confusion.

2.3 Summary

Table 1 presents a comparison of the tools described in this chapter. Each lines represents a feature, showing whether the tool has (y) or does not have (n) a certain feature. When a certain feature depends on another, it is indented in a tree-like fashion with ">>". In this case, if the tool does not have the "parent" feature, it is marked with an "x". Empty cells means that the features were not evaluated.

Table 1: Software comparison.

	Microsoft Visio	Axure RP Pro	Denim	SketchiXML	Balsamiq	CogTool
Free	n	n	y	y	n	y
UI-prototype exclusive (vs generic diagrammatic tool)	n	y	y	y	y	n
UI components for multiple environments (vs web-page prototype only)	y	y	n	y	y	y
Drawing widgets	n	n	n	y	n	n
>> Evolution of widgets	x	x	x	n	x	x
Element manipulation	y		n	n	y	y
Undo/Redo	y		y	y	y	y
Group/Ungroup	y		y	n	y	n
>> Select internal objects	y		x	x	n	x
Cut/Copy/Paste	y		y	y	y	y
>> Copies the action	n		n	n	y	n
Zoom levels	y		y	y	y	y
Guidelines	n		n	n	y	n
Layer ordering	y		n	n	y	y
Sketchy visual	n	n	y	y	y	y
Actions	n	y	y	y	y	y
>> Beyond navigation	x	y	n	n	n	n
>> Sketchy interaction	x	n	y	y	n	y
>> Event	x	y	y	n	n	y
>> Conditions	x	n	y	n	n	n
Prototype evaluation	x	y	y	n	y	y
Save	y	y	y	n	y	y

As we will see in the next chapters, UISKEI targets the prototyping process with a pen-based interaction approach, not only during interface building but also when defining the interface behavior. Moreover, the behavior defined should go beyond navigational purposes and be conditionally triggered, combination only present in Axure RP Pro, but without the sketchy interaction.