10
Bibliography

Alferes, J., Banti, F., & Brogi, A. (2006). An Event-Condition-Action Logic Programming Language. *Proceedings of the 10 th European Conference on Logics in Artificial Intelligence* (JELIA ‘06), pp. 29-42.

Bailey, J., Poulovassilis, A., & Wood, P. T. (2002). An event-condition-action language for XML. *Proceedings of the 11th international conference on World Wide Web* (WWW '02), pp. 486-495.

Barber, G. (2009, March 25). *16 Design Tools for Prototyping and Wireframing*. Retrieved July 22, 2010, from sitepoint: http://articles.sitepoint.com/article/tools-prototyping-wireframing

Callahan, J., Hopkins, D., Weiser, M., & Shneiderman, B. (1988). An empirical comparison of pie vs. linear menus. *Proceedings of the SIGCHI conference on Human factors in computing systems* (CHI '88), pp. 95-100.

Carnegie Mellon University. (2009). *CogTool*. Retrieved July 28, 2010, from Human-Computer Interaction Institute: http://cogtool.hcii.cs.cmu.edu/

Cha, S.-H., Shin, Y.-C., & Srihari, S. N. (1999). Approximate Stroke Sequence String Matching Algorithm for Character Recognition and Analysis. *Proceedings of the Fifth International Conference on Document Analysis and Recognition* (ICDAR '99), p. 53.

Coyette, A., Faulkner, S., Kolp, M., Limbourg, Q., & Vanderdonckt, J. (2004). SketchiXML: towards a multi-agent design tool for sketching user interfaces based on USIXML. *Proceedings of the 3rd annual conference on Task models and diagrams* , pp. 75-82.

Coyette, A., Schimke, S., Vanderdonckt, J., & Vielhauer, C. (2007). Trainable sketch recognizer for graphical user interface design. *Proceedings of the 11th IFIP TC 13 international conference on Human-computer interaction* (INTERACT'07), pp. 124-135.

Dahlbäck, N., Jönsson, A., & Ahrenberg, L. (1993). Wizard of Oz studies: why and how. *Proceedings of the 1st international conference on Intelligent user interfaces* (IUI '93), pp. 193-200.

Davis, R. C., Saponas, T. S., Shilman, M., & Landay, J. A. (2007). SketchWizard: Wizard of Oz prototyping of pen-based user interfaces. *Proceedings of the 20th* *annual ACM symposium on User interface software and technology* (UIST '07), pp. 119-128.

Douglas, D. H., & Peucker, T. K. (1973, December). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization , Volume 10, Number 2*, pp. 112-122.

Frye, J., & Franke, B. (2008). PDP: Pen Driven Programming. *Proceedings of the 22nd British HCI Group Annual Conference on People and Computers: Culture, Creativity, Interaction - Volume 2* , pp. 127-130.

Gross, M. D. (2009). Visual languages and visual thinking: sketch based interaction and modeling. *SBIM '09: Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling* (pp. 7-11). New Orleans, Louisiana: ACM.

Gusfield, D. (1997). *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology.* Cambridge University Press.

Hammond, T. A. (2009). IUI'09 workshop summary: sketch recognition. *Proceedings of the 13th international conference on Intelligent user interfaces* (pp. 501-502). Sanibel Island, Florida, USA: ACM.

Hammond, T., & Davis, R. (2006). LADDER: a language to describe drawing, display, and editing in sketch recognition. *ACM SIGGRAPH 2006 Courses* (SIGGRAPH '06).

Hammond, T., Eoff, B., Paulson, B., Wolin, A., Dahmen, K., Johnston, J., et al. (2008). Free-sketch recognition: putting the chi in sketching. *CHI '08 extended abstracts on Human factors in computing systems* (CHI EA '08), pp. 3027-3032.

Hammond, T., Lank, E., & Adler, A. (2010). SkCHI: designing sketch recognition interfaces. *CHI EA '10: Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems* (pp. 4501-4504). Atlanta, Georgia, USA: ACM.

Harrelson, D. (2009, March 24). *Rapid Prototyping Tools*. Retrieved July 22, 2010, from Adaptive Path Blog: http://www.adaptivepath.com/blog/2009/03/24/rapid-prototyping-tools/

Hong, J., Landay, J., Long, A. C., & Mankoff, J. (2002). *Sketch Recognizers from the End-User's, the Designer's, and the Programmer's Perspective.*

Hundhausen, C. D., Balkar, A., Nuur, M., & Trent, S. (2007). WOZ pro: a pen-based low fidelity prototyping environment to support wizard of oz studies. *CHI '07 extended abstracts on Human factors in computing systems* (CHI EA '07), pp. 2453-2458.

Kieffer, S., Coyette, A., & Vanderdonckt, J. (2010). User interface design by sketching: a complexity analysis of widget representations. *EICS '10: Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems* (pp. 57-66). Berlin, Germany: ACM.

Kurtenbach, G. (2010). Pen-based computing. *XRDS , 16* (4), 14-20.

Landay, J. A., & Myers, B. A. (1995). Interactive sketching for the early stages of user interface design. *Proceedings of the SIGCHI conference on Human factors in computing systems* , pp. 43-50.

Landay, J. A., & Myers, B. A. (2001). Sketching Interfaces: Toward More Human Interface Design. *Computer , 34*, 56-64.

Lazar, J., Feng, J. H., & Hochheiser, H. (2010). *Research Methods in Human-Computer Interaction.* John Wiley & Sons Ltd.

Lin, J., Thomsen, M., & Landay, J. A. (2002). A visual language for sketching large and complex interactive designs. *Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves* , pp. 307-314.

Paulson, B., & Hammond, T. (2008). PaleoSketch: accurate primitive sketch recognition and beautification. *Proceedings of the 13th international conference on Intelligent user interfaces* (IUI '08), pp. 1-10.

Po, B. A., Fisher, B. D., & Booth, K. S. (2005). Comparing cursor orientations for mouse, pointer, and pen interaction. *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 291-300). Portland, Oregon, USA: ACM.

Preece, J., Rogers, Y., & Sharp, H. (2002). *Interaction design: beyond human-computer interaction.* John Wiley.

Schimke, S., & Vielhauer, C. (2007). Similarity searching for on-line handwritten documents. *Journal on Multimodal User Interfaces , 1* (2), 49-54.

Schmucker, K. J. (1996). Rapid prototyping using visual programming tools. *Conference companion on Human factors in computing systems: common ground* , pp. 359-360.

Segura, V. C., & Barbosa, S. D. (2008). *Ferramenta de apoio ao esboço de interfaces gráficas através da interação com caneta.* Relatório de Projeto Final, PUC-Rio, Departamento de Informática.

Segura, V. C., & Barbosa, S. D. (2009). UISK: Supporting Model-Driven and Sketch-Driven Paperless Prototyping. *Proceedings of the 13th International Conference on Human-Computer Interaction. Part I: New Trends* , pp. 697--705.

Sezgin, T. M., Stahovich, T., & Davis, R. (2006). Sketch based interfaces: early processing for sketch understanding. *ACM SIGGRAPH 2006 Courses* (SIGGRAPH '06).

Szekely, P. A. (1994). User Interface Prototyping: Tools and Techniques. *Proceedings of the Workshop on Software Engineering and Human-Computer Interaction* (ICSE '94), pp. 76-92.

Tandler, P., & Prante, T. (2001, November). Using Incremental Gesture Recognition to Provide Immediate Feedback while Drawing Pen Gestures. *14th Annual ACM Symposium on User Interface Software and Technology* (UIST 2001).

Tohidi, M., Buxton, W., Baecker, R., & Sellen, A. (2006a). Getting the right design and the design right. *Proceedings of the SIGCHI conference on Human Factors in computing systems* (CHI '06), pp. 1243-1252.

Tohidi, M., Buxton, W., Baecker, R., & Sellen, A. (2006b). User sketches: a quick, inexpensive, and effective way to elicit more reflective user feedback. *Proceedings of the 4th Nordic conference on Human-computer interaction: changing roles* (NordiCHI '06), pp. 105-114.

11 Appendix A: Implementing UISKEI

We developed UISKEI in C#, using Microsoft Visual Studio 2007 and the .NET Framework 3.5. It consists of three main projects — `uskModel`, `uskRecognizer` and `uskWizard` — which will be presented in the following sections.

11.1 uskModel

This project contains all the core data and logics of an UISKEI's project. All the project's information that needs to be saved to a file is in this project, so many of its classes implement the C# interface `ISerializable`, allowing the serialization of necessary data to a binary file. The class diagram of this project can be seen in Figure 33.

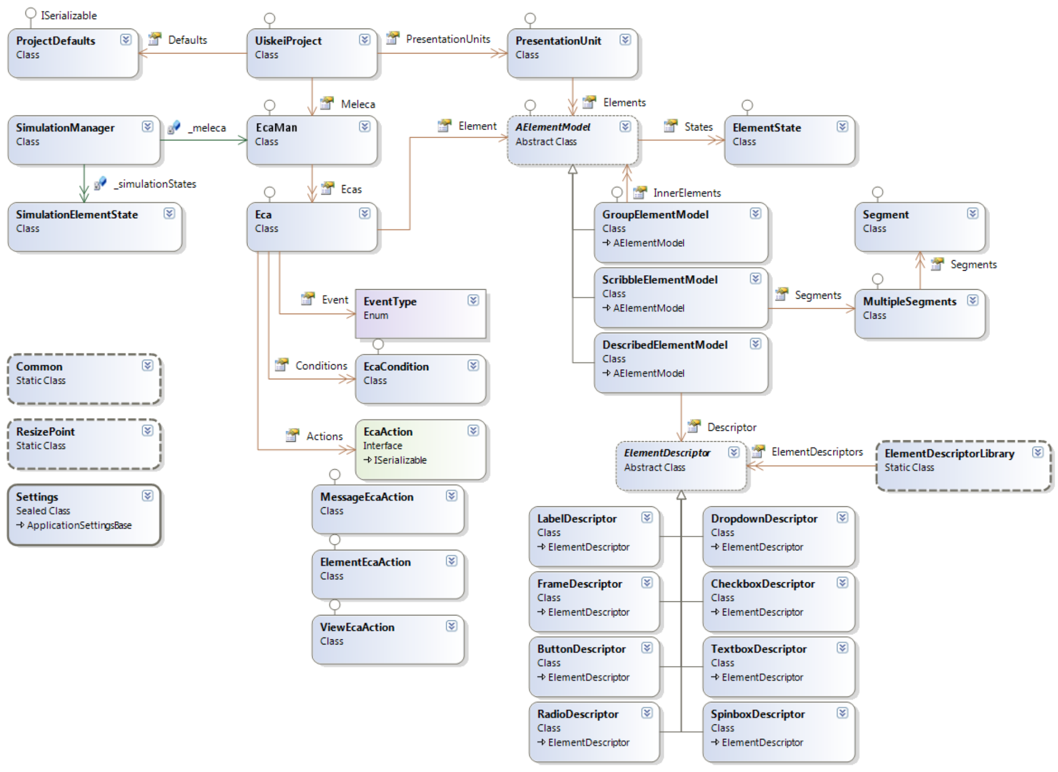

ISerializable
ProjectDefaults Class
Defaults
UiskeiProject Class
PresentationUnits
PresentationUnit Class
Meleca
Elements
SimulationManager Class
_meleca
EcaMan Class
Element
AElementModel Abstract Class
States
ElementState Class
_simulationStates
Ecas
InnerElements
SimulationElementState Class
Eca Class
GroupElementModel Class → AElementModel
Segment Class
Segments
ScribbleElementModel Class → AElementModel
Segments
MultipleSegments Class
Event
EventType Enum
DescribedElementModel Class → AElementModel
Common Static Class
Conditions
EcaCondition Class
ResizePoint Static Class
Actions
EcaAction Interface → ISerializable
Descriptor
ElementDescriptor Abstract Class
ElementDescriptors
ElementDescriptorLibrary Static Class
Settings Sealed Class → ApplicationSettingsBase
MessageEcaAction Class
LabelDescriptor Class → ElementDescriptor
DropdownDescriptor Class → ElementDescriptor
ElementEcaAction Class
FrameDescriptor Class → ElementDescriptor
CheckboxDescriptor Class → ElementDescriptor
ViewEcaAction Class
ButtonDescriptor Class → ElementDescriptor
TextboxDescriptor Class → ElementDescriptor
RadioDescriptor Class → ElementDescriptor
SpinboxDescriptor Class → ElementDescriptor

Figure 33: Class diagram for uskModel.

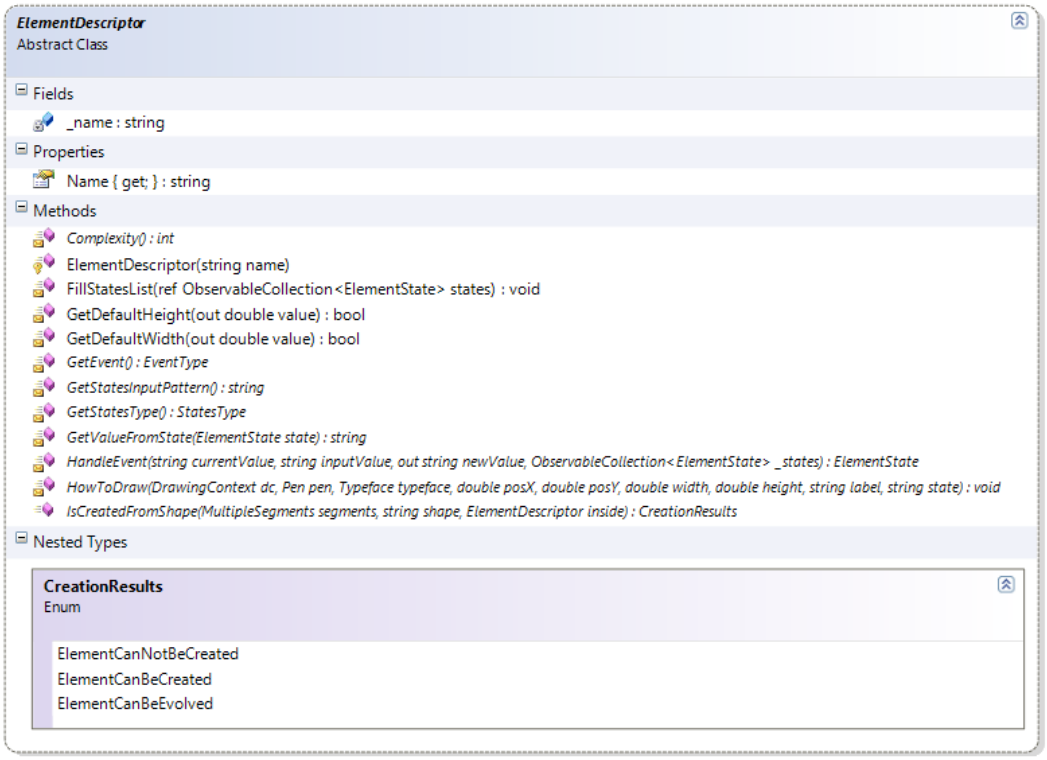
The main class of uskModel is UiskeiProject, which stores a list of PresentationUnits, a ProjectDefaults and a ECAMan. A PresentationUnit stores a list of abstract elements, AElementModel. For most uses, the abstract element is sufficient: we only need to know the "concrete" element to create the correct visualization and to create the group of selected elements. All the other operations are done with abstract elements.

An element can contain a list of ElementStates, which contains a name and may have additional parameters that are interpreted accordingly. For example, the textbox's states discussed in Section 4.5 have the pattern and the sample text as additional parameters.

The AElementModel is the base class for three different concrete classes:

- GroupElementModel → represents a group of elements, therefore it has a list of AElementModels.
- ScribbleElementModel → represents the unidentified drawing, therefore containing the drawing converted to a MultipleSegments, composed of a list of Segments and a bounding box.
- DescribedElementModel → represents the identified drawing, which was converted to a widget. It contains a reference to the ElementDescriptor which describes the widget.

As previously discussed in Section 4.3, we implemented the descriptors hard-coded, so it is possible to see in the class diagram all the inheritance of classes and the available elements. Besides that, the remaining code is already prepared to handle generic descriptors, since the "concrete descriptors" are private to this project and never referenced. The only class that uses the "concrete descriptors" is the loader, ElementDescriptorLibrary. When the descriptor language is available, we will only have to change the loader to create descriptor instances from the language. This descriptor language should handle all the necessary information, currently coded in the ElementDescriptors, as shown in Figure 34.



ElementDescriptor
Abstract Class

Fields
_name : string

Properties
Name { get; } : string

Methods
Complexity() : int
ElementDescriptor(string name)
FillStatesList(ref ObservableCollection<ElementState> states) : void
GetDefaultHeight(out double value) : bool
GetDefaultWidth(out double value) : bool
GetEvent() : EventType
GetStatesInputPattern() : string
GetStatesType() : StatesType
GetValueFromState(ElementState state) : string
HandleEvent(string currentValue, string inputValue, out string newValue, ObservableCollection<ElementState> _states) : ElementState
HowToDraw(DrawingContext dc, Pen pen, Typeface typeface, double posX, double posY, double width, double height, string label, string state) : void
IsCreatedFromShape(MultipleSegments segments, string shape, ElementDescriptor inside) : CreationResults

Nested Types

CreationResults
Enum

ElementCanNotBeCreated
ElementCanBeCreated
ElementCanBeEvolved

Figure 34: ElementDescriptor class.

The `ProjectDefaults` stores some default values for the project, such as a default width and height for newly created presentation units. The `ECAMan` stores all available `ECAs` for the project in a list, organized by the element which triggers the event. An `ECA` stores the `EventType` which triggers the behavior, the list of `EcaConditions` to be tested and the list of `EcaActions` to be performed.

Since the available ECA conditions are only related to elements, the `EcaCondition` class has a default constructor with two parameters: an `AElementModel` and a `TestType`. `TestType` is a nested enumeration of the available test operations, listed in Section 5.1.2.

In the other hand, there are three different types of ECA actions, so the `EcaAction` was implemented as an interface. The `MessageEcaAction`, `ElementEcaAction` and `ViewEcaAction` implement this interface. All three classes have a single constructor which receives the target object (a presentation unit in the case of `MessageEcaAction`, an element in the case of `ElementEcaAction` and a string in the case of `ViewEcaAction`) and an `ActionType`. Similar to the `TestType`, `ActionType` is a nested enumerator in each class of available operations, as listed in Section 5.1.3. An overview of the `TestType` and `ActionType` can be seen in Figure 35.

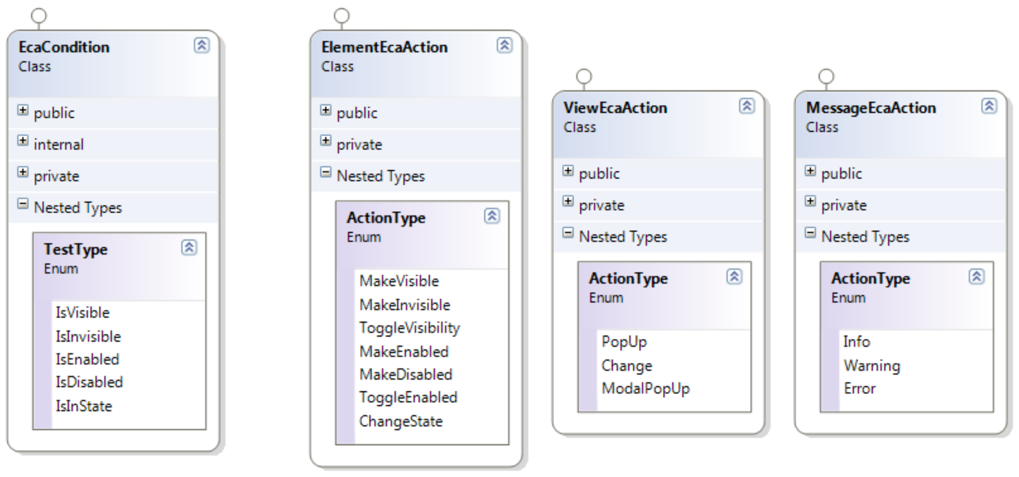


Figure 35: Operations enums.

Also in `uskModel` there is the `SimulationManager`, which references a `ECAMan`. This class controls the state of the elements — their properties and `ElementState` — during a prototype evaluation session, creating a data structure `SimulationElementState` to store it. An ECA is activated through the `SimulationManager`, testing its conditions and executing its actions according to the `SimulationElementStates`. As discussed in Section 6, since the manager only references the `ECAMan`, changes in ECAs are reflected on-the-fly during the prototype evaluation session.

The `Common` class contains a list of common methods used in various projects. The `ResizePoint` class represents the eight handles or points that appears onscreen when the user tries to resize an element. Since depending on the manipulated point the resize result is different, this is a parameter of the resize method of an element. When no `ResizePoint` is specified, it is considered that the resize occurs in the `SouthEast` direction.

11.2 uskRecognizer

This project is the one responsible for recognizing the user's drawings. It handles the `MultipleSegments` and `Segment` data structures and evaluates the Shape associated to it. The classes that compose this project can be seen in Figure 36.

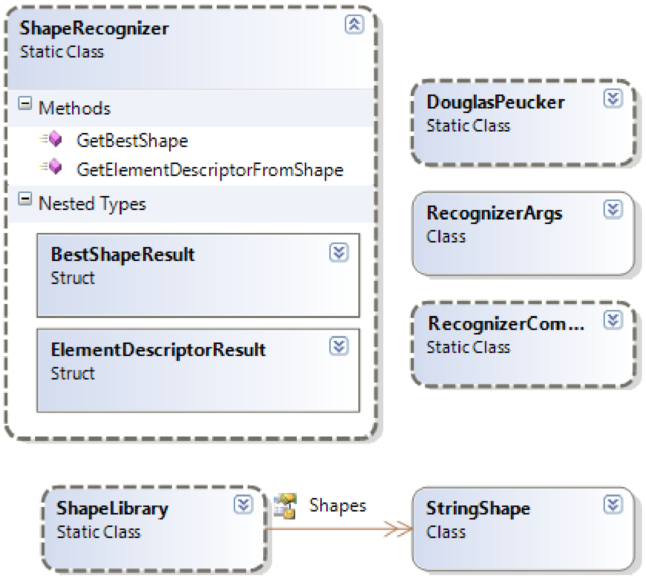


Figure 36: Class diagram for uskRecognizer.

The main exported class is the ShapeRecognizer. It has two methods, GetBestShape and GetElementDescriptorFromShape. The first method receives the MultipleSegments to be recognized and returns the structure BestShapeResult, which references the shape with the least distance and the distance value.

After associating the MultipleSegments to a shape, the second method is called. It receives the MultipleSegments, the BestShapeResult and the element in which the drawing was made (or null otherwise). It returns a data structure ElementDescriptorResult, which contains the descriptor (or null if not recognized) and how the element can be created (if it is a new element or an evolved element).

The recognition process was divided in two methods for two reasons. First, for test purposes, since the association to a shape was dissociated from the recognition as an element. Second because since several drawings may enter the recognition process, the association to a shape can happen only one time and the element recognition may happen iteratively. For example, if the user draws several rectangles in sequence and in the end draws a line in the first one, the first one should be converted into a textbox and the others, to a button. So, the association to a shape may happen only in the beginning (several rectangles and a horizontal line) but the element recognition must happen iteratively (so the rectangle is first recognized as a button and later as a textbox).

The `GetBestShape` method uses the `ShapeLibrary` to go through the list of loaded shapes. A `StringShape` is defined as explained in Section 4.2 and loaded from a text file with a `.shp` extension.

The other classes are internal for the project. `RecognizerCommon` contains common methods used in the project. `RecognizerArgs` is a structure that associates a `Segment` to its description as a string of directions and is used during the association to a shape step. Finally, the `DouglasPeucker` class is responsible for the Douglas-Peucker algorithm of simplifying line segments as discussed in Section 4.4.

There are a number of variations related to the recognition process that needs to be defined. The Douglas-Peucker simplification algorithm may or may not be used and, if used, we need to establish a value for tolerance. Then, when the segment is being converted to a string, we need to define how far apart the points must be in order to be converted into a character direction. Finally, with the segment string, we need to define which will be the costs used in the Levenshtein edit distance algorithm. An experimental test to determine these variations is detailed in Chapter 7.

11.3 uskWizard

The last project is `uskWizard`. It contains the graphical user interface of UISKEI, entirely built using XAML. The project's classes can be seen in Figure 37.

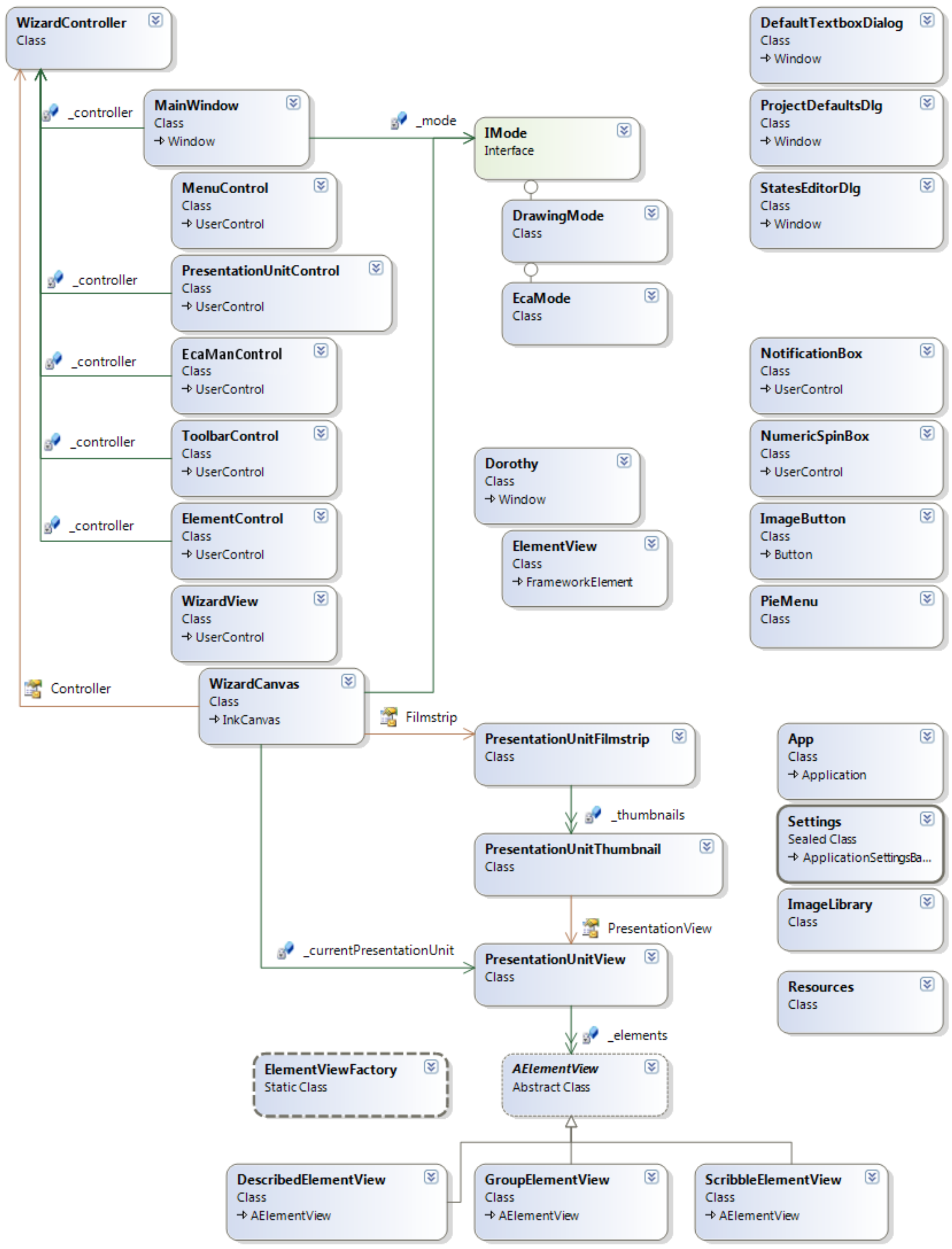

WizardController Class
_controller
MainWindow Class Window
_mode
IMode Interface
MenuControl Class UserControl
DrawingMode Class
_controller
PresentationUnitControl Class UserControl
EcaMode Class
_controller
EcaManControl Class UserControl
_controller
ToolbarControl Class UserControl
_controller
ElementControl Class UserControl
WizardView Class UserControl
Dorothy Class Window
ElementView Class FrameworkElement
Controller
WizardCanvas Class InkCanvas
Filmstrip
PresentationUnitFilmstrip Class
_thumbnails
PresentationUnitThumbnail Class
PresentationView
_currentPresentationUnit
PresentationUnitView Class
_elements
ElementViewFactory Static Class
AElementView Abstract Class
DescribedElementView Class AElementView
GroupElementView Class AElementView
ScribbleElementView Class AElementView
DefaultTextboxDialog Class Window
ProjectDefaultsDlg Class Window
StatesEditorDlg Class Window
NotificationBox Class UserControl
NumericSpinBox Class UserControl
ImageButton Class Button
PieMenu Class
App Class Application
Settings Sealed Class ApplicationSettingsBa...
ImageLibrary Class
Resources Class

Figure 37: Class diagram for uskWizard.

The `MainWindow` is composed by several controls — `MenuControl`, `PresentationUnitControl`, `ECAManControl`, `ToolbarControl`, `ElementControl` — and the `WizardView`, a `UserControl` with a `WizardCanvas`. Since the diagram generated by Visual Studio lacks the representation of GUI composition, the classes were displayed in a tree-like fashion in Figure 37 to illustrate this composition.

Each control references the same `WizardController`, object that keeps the current state of the interface, for example, the current project, the current presentation unit, the selected elements, etc. Since all controls references the same object, they are kept in sync.

The `WizardCanvas` is where the presentation units are displayed and the pen-based interaction occurs. It contains a `PresentationUnitFilmstrip` (shown in Figure 25), composed of a list of thumbnails, `PresentationUnitThumbnails`. Each thumbnail contains a scaled `PresentationUnitView`, which is the representation of a presentation unit. Each `PresentationUnitView` contains a list of `AElementViews`, which, similarly, are the representation of an `AElementModel`. The `ElementViewFactory` is responsible for creating the correct concrete implementation of `AElementView`.

This architecture does not use the `FrameworkElement` paradigm, so we had to explicitly handle the `Draw` event and also the mouse events (e.g. click, enter, leave), having total control of how it worked. Also, since there is no reference to parent objects, we were able to use the same `PresentationUnitView` in the filmstrip and in the canvas, reducing the number of objects in memory.

The `MainWindow` also controls the current mode of UISKEI, between four possible modes: drawing mode, recognition mode, ECA mode and simulation mode. The first three modes changes how the designer interacts with the canvas, while the last one creates a new dialog.

The drawing mode and the recognition mode are essentially the same, only having to recognize the drawings or not. So, the `DrawingMode` class is used for both and a `boolean` passed as a parameter in the constructor sets the recognition process on or off. The ECA mode is handled by the `EcaMode` class.

The modes that changes the interaction with the canvas are abstracted with the `IMode` interface. When an `IMode` is set on the canvas, the previous one is deactivated and the new one is activated, so the modes can connect/disconnect to the canvas' events needed. For example, for the `DrawingMode`, the `StrokeCollected` event is important and the `MouseMove` is not, while for the `EcaMode` is the opposite: the `MouseMove` event is important and the `StrokeCollected` is not.

The last mode, simulation mode, displays a new window with the interactive prototype for evaluation purposes. Since one of the future works idea is to have a Wizard of Oz approach to the simulation (as will be discussed in Section 9.4), the simulation user dialog was called `Dorothy` and it is composed by `ElementViews`. To investigate how the `FrameworkElement` paradigm works, the `ElementView` inherits from `FrameworkElement`.

Other smaller dialogs also are part of `uskWizard`. The `DefaultTextboxDialog` is a standard dialog with a instructions message and a textbox, used to input the text of a message action. The `ProjectDefaultsDlg` configures the `uskModel.ProjectDefaults` of the current project. The `StatesEditorDlg` is the dialog to edit element's states. By the time, it contains only a textbox, but it can be improved to show a data grid with on-the-fly validation, for example.

12
Appendix B: Evaluation study script

The test comparing the three prototyping techniques took place in the beginning of February / 2011 and recruited 12 participants. To each one was presented the script below, freely translated from Portuguese (the participants' native language), keeping its format, organization and structure.

12.1
Description

This study aims to compare three different GUI prototyping techniques (paper prototyping, prototyping using UISKEI and prototyping using Balsamiq), evaluating how they can be applied to a use case described through the test. It is divided in stages, described below:

- Initial stage → Only a questionnaire about your familiarity with the use of computers and prototyping tools.
- 1st and 2nd cycles → Each one is further divided in three parts:
    - Scenario → A story to motivate the tool usage.
    - Task → The activity to be done related to the scenario, using all three techniques. In the first cycle, a video will be show to introduce you to the tools.
    - Questionnaire → Three questionnaires will be presented in each cycle, one after the use of each tool during task execution, to obtain your opinion about each technique.
- Final cycle → An additional scenario will be shown, but the task will not need to be performed: we will only talk about how you would perform it. After the discussion, the same three questionnaires should be filled and a quick interview will take place.

If you wish to be a test participant, in order to guarantee your anonymity and privacy rights, and to assure the adequate use of the collected data, you will need to sign an informed consent form.

IMPORTANT: What is being evaluated are the tools, **NOT** you. There is not a "right way" to perform each task. The difficulties and facilities that each tool provide in each task are the main focus of the test.

12.2 Questionnaire

1) How often do you use computers?
   - ☐ 4 - Many times a day
   - ☐ 3 - At least once a day
   - ☐ 2 - At least once a week
   - ☐ 1 - Once in a while
   - ☐ 0 - Never
2) Do you know any programming language?
   - ☐ 0 - No (go to question 3)
   - ☐ 1 - Yes, Which ones? ____

   a) With which ones are you most familiarized? ________________________

   b) How often do you use this programming language?
      - ☐ 4 - Many times a day
      - ☐ 3 - At least once a day
      - ☐ 2 - At least once a week
      - ☐ 1 - Once in a while

   c) How would you classify your knowledge about this programming language?
      - ☐ 4 - I know the language deeply and I can do everything I want with it
      - ☐ 3 - I have a fair knowledge of the language, but sometimes I need to learn a little more
      - ☐ 2 - I know little about the language, knowing only its basic usage
      - ☐ 1 - Still learning the basics

3) Have you already had to design a graphical user interface (an application window, a web page, for example), i.e., plan how the interface elements are arranged and how the interface "works"?

   ☐ 0 - No (end of questionnaire)

   ☐ 1 - Yes

   a) How often have you had to make an interface design?

      ☐ 3 - Practically every week

      ☐ 2 - Sometimes a month

      ☐ 1 - Sometimes a year

   b) How do you usually elaborate the designs? (you may mark more than one option)

      ☐ Using paper drawings

      ☐ Using a generic tool

         ☐ Image editing software (Paint, Photoshop, GIMP, etc)

         ☐ Power Point

         ☐ HTML

         ☐ Others? ______________________________

      ☐ Using a specific prototyping tool

         ☐ Visio

         ☐ Balsamiq

         ☐ Dreamweaver

         ☐ Specific language designer (Qt Designer, Expression Blend, etc)

         ☐ Others? ______________________________

      ☐ Implementing in code

4) Have you ever used any of the prototyping tools to be evaluated in this test?

   ☐ 0 - No (end of questionnaire)

   ☐ 1 - Yes, please fill in the following table:

	Paper	Balsamiq	UISKEI
How would you classify your knowledge of this tool? 4 - I know the tool deeply and I can do everything I want with it 3 - I have a fair knowledge of the tool, but sometimes I need to learn a little more 2 - I know little about the tool, knowing only its basic usage 1 - Still learning the basics 0 - I don't know / Never used			
How often do you use this tool? 4 - Many times a day 3 - At least once a day 2 - At least once a week 1 - Once in a while 0 - Never			

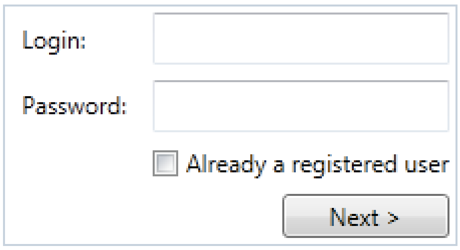
12.3 First Cycle

12.3.1. Scenario 01

You are developing an e-commerce application and wish to plan how the checkout process will occur after the products are in the cart. You imagine a system where the user must identify him/herself to the system (entering a username and a password, for example), informing whether he/she is already a registered user. If he/she is already a registered user, he/she will see the "Checkout of registered user" screen, where he/she can choose the payment information (as address and credit card) previously used. If he/she is not a registered user, he/she will see the "Register user" screen, in which he/she must enter other information, as e-mail, password confirmation, etc.

12.3.2. Task 01

Sketch the screen below, related to the presented scenario, in the available three tools and in the established order, as you would present the planned prototype to a client.



After drawing the screens, simulate the prototype, exploring all the alternative interaction paths.

Ps.: The resulting screens ("Checkout of registered user" and "Register user") don't need to be sketched. They are only needed to distinguish which would be the resulting screen when pressing the button.

12.4 Second Cycle

12.4.1. Scenario 02

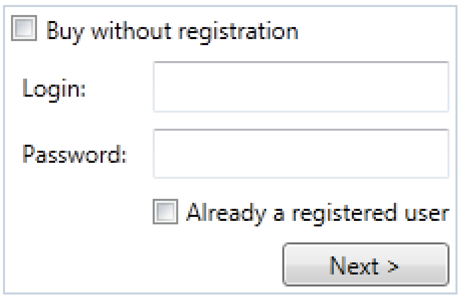
Talking to a friend of yours, he shows himself outraged for being obliged to make a registration in a web store where he plans to never buy again. To avoid this kind of dissatisfaction of your future clients, you plan to add to that initial screen an option of "Buy without registration". If the client activates this option, the client identification fields (login and password) will be disabled. Depending of the chosen options, the client could go to different screens, described below:

"Buy without registration"	**"Already a registered user"**	**Result**
☐	☐	"Register user"
	☑	"Checkout of registered user"
☑	☐	"Checkout without register"
	☑	"Error 1" (client informed that doesn't want to buy with a registration and that he/she is a registered user)

12.4.2. Task 02

Sketch the screen below, related to the presented scenario, in the available three tools and in the established order. Then, demonstrate how you would test the prototype with a user, showing how you would do to display that the checkbox

"Buy without registration" determines whether the fields "Login" and "Password" are enabled/disabled and how the combination of this checkbox with the other one can lead to 4 different results.


☐ Buy without registration

Login:

Password:

☐ Already a registered user

Next >

12.5 Third Cycle

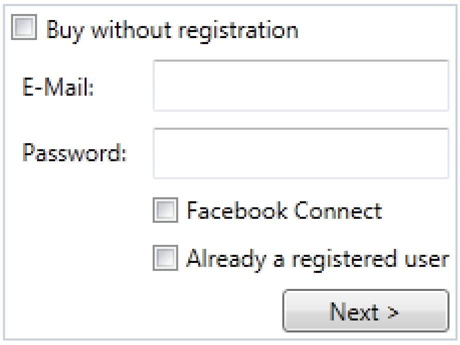
12.5.1. Scenario 03

Navigating in competitors websites, you discover that there is a "Facebook connect" feature, allowing a site to use the user's Facebook registration to identify him/herself in other sites. To integrate this possibility into your website, you change the text from "Login" to "E-Mail" and add a new option to indicate whether the information provided is from the site or from the Facebook. With this new option, the possible paths are:

"Facebook Connect"	**"Buy without registration"**	**"Already a registered user"**	**Result**
☐	☐	☐	"Register user"
		☑	"Checkout of registered user"
	☑	☐	"Checkout without register"
		☑	"Error 1" (Without registration X Registered user)
☑	☐	☐	"Register user using Facebook"
		☑	"Checkout of Facebook registered user"
	☑	☐	"Error 2" (Without registration X Facebook)
		☑	"Error 3" (Without registration X Registered with Facebook)

12.5.2. Task 03

Talk about what modification you would make in the prototypes and how you would test them, using the available three tools and in the established order. Try talking about how each tool makes the process of building the prototype and its behavior easier or harder.



13
Appendix B: Complete test results

The tables in this appendix show the data collected from the evaluation study for every participant, represented by `p<n>` (the fifth participant is named `p5`, for example). The following table shows the tool presentation order, where `P` stands for paper, `B` stands for Balsamiq and `U` stands for UISKEI. It is possible to see that every combination (6 in total) was performed with two different participants.

Table 9: Tools presentation order.

		p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12
Order	**P**	1	1	2	3	2	3	1	3	1	2	3	2
	B	2	3	1	2	3	1	3	1	2	1	2	3
	U	3	2	3	1	1	2	2	2	3	3	1	1

The next table shows the answers to the initial questionnaire presented to the participants. The complete questionnaire script and the answer code can be seen in Section 12.1.

Table 10: Questionnaire answers.

		p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12
Questionnaire	**1**	4	4	4	4	4	4	4	4	4	4	4	4
	2	1	1	1	1	1	1	1	1	1	1	1	1
	2b	2	4	1	1	4	3	2	4	3	2	4	4
	2c	2	3	2	2	3	3	3	4	4	2	3	3
	3	0	1	1	1	1	1	1	1	1	1	1	1
	3a	x	1	2	1	1	3	1	1	2	2	1	3
	4	1	0	1	0	0	1	1	1	1	0	0	1
	4p1	2	x	2	x	x	3	0	2	4	x	x	4
	4p2	0	x	1	x	x	1	0	1	1	x	x	1
	4b1	0	x	0	x	x	0	0	0	0	x	x	0
	4b2	0	x	0	x	x	0	0	0	0	x	x	0
	4u1	1	x	0	x	x	2	2	0	0	x	x	0
	4u2	0	x	0	x	x	1	1	0	0	x	x	0

The following tables show the scores given by the participants in the questionnaire presented after each cycle (the questions can be found in Section 8.1). Also, they present the average score and standard deviation for each question and group of questions (grouped in user interface questions and interaction questions, as discussed in Section 8.2).

Table 11: First cycle questionnaire answers and statistics.

			p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	Ave	Std Dev	Ave	Std Dev
1st Cycle	Paper	1	5	5	5	5	5	5	5	3	5	5	5	5	4.83	0.58	4.33	0.86
		2	5	4	4	5	3	5	4	5	4	4	4	5	4.33	0.65		
		3	5	3	5	5	4	5	3	5	5	2	4	5	4.25	1.06		
		4	5	5	5	4	2	4	4	4	3	4	3	4	3.92	0.90		
		5	3	2	5	5	4	5	4	5	3	4	5	5	4.17	1.03	3.74	1.22
		6	2	4	5	5	2	5	4	2	4	3	3	1	3.33	1.37		
		7	5	5	5	5	4	5	4	2	4	3	3	4	4.08	1.00		
		8	5	5	5	5	3	5	4	3	5	5	4	4	4.42	0.79		
		9	3	3	5	4	2	3	4	2	3	4	3	1	3.08	1.08		
		10	5	3	5	3	2	5	3	1	3	4	5	1	3.33	1.50		
	Balsamiq	1	4	3	4	5	5	4	5	5	5	4	5	4	4.42	0.67	4.25	0.89
		2	3	4	5	5	3	5	5	5	5	2	5	4	4.25	1.06		
		3	3	5	4	5	2	5	5	5	4	4	5	5	4.33	0.98		
		4	3	4	4	5	2	4	5	4	4	4	5	4	4.00	0.85		
		5	2	2	5	3	3	4	2	2	2	3	4	3	2.92	1.00	2.97	1.10
		6	1	3	2	4	2	2	3	3	3	1	2	2	2.33	0.89		
		7	2	4	5	3	2	3	4	5	4	2	3	4	3.42	1.08		
		8	2	4	4	4	4	4	4	4	4	3	3	3	3.58	0.67		
		9	2	3	4	3	2	2	4	2	2	1	1	3	2.42	1.00		
		10	4	3	5	5	2	4	4	4	3	1	2	1	3.17	1.40		
	UISKEI	1	5	3	3	5	2	4	4	4	5	3	5	4	3.92	1.00	4.02	1.00
		2	5	3	2	5	3	3	3	4	5	2	5	3	3.58	1.16		
		3	5	4	4	5	5	5	4	5	3	4	4	5	4.42	0.67		
		4	5	5	2	5	3	3	4	5	5	4	5	4	4.17	1.03		
		5	2	3	3	5	4	5	2	5	4	4	5	5	3.92	1.16	3.94	1.06
		6	4	4	3	5	5	5	3	5	5	3	5	4	4.25	0.87		
		7	4	3	3	4	3	5	2	4	4	3	5	4	3.67	0.89		
		8	4	3	2	3	4	5	2	5	5	2	5	5	3.75	1.29		
		9	5	4	2	5	3	4	3	5	5	3	5	4	4.00	1.04		
		10	5	5	1	5	4	5	4	3	4	4	5	4	4.08	1.16		

Table 12: Second cycle questionnaire answers and statistics.

			p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	Ave	Std Dev	Ave	Std Dev
2nd Cycle	Paper	1	5	5	5	5	5	4	5	3	4	5	5	5	4.67	0.65	3.83	1.10
		2	3	5	3	5	3	5	3	2	5	4	3	5	3.83	1.11		
		3	3	4	2	5	2	5	3	5	4	1	4	4	3.50	1.31		
		4	3	3	5	4	3	4	4	3	3	3	2	3	3.33	0.78		
		5	5	2	5	5	3	5	4	4	2	4	5	5	4.08	1.16	3.35	1.34
		6	2	2	4	5	2	5	3	2	4	3	2	1	2.92	1.31		
		7	4	4	4	5	4	5	4	1	4	5	3	3	3.83	1.11		
		8	4	4	3	5	3	5	3	2	3	5	3	1	3.42	1.24		
		9	3	2	5	4	2	4	4	1	3	5	2	1	3.00	1.41		
		10	5	2	4	4	2	4	3	1	2	1	5	1	2.83	1.53		
	Balsamiq	1	5	5	5	5	4	5	5	5	5	4	5	4	4.75	0.45	4.31	0.90
		2	4	5	5	5	3	5	3	5	4	4	5	3	4.25	0.87		
		3	5	5	5	5	2	5	5	4	4	4	5	2	4.25	1.14		
		4	2	4	5	5	4	4	5	4	3	4	5	3	4.00	0.95		
		5	5	2	5	5	4	4	5	3	4	4	4	5	4.17	0.94	3.35	1.42
		6	1	3	5	5	2	4	4	2	1	2	2	1	2.67	1.50		
		7	4	5	5	5	4	5	4	4	3	5	2	4	4.17	0.94		
		8	4	4	5	5	2	5	3	5	2	4	2	1	3.50	1.45		
		9	2	2	5	4	2	3	5	1	2	1	1	1	2.42	1.51		
		10	2	3	5	5	3	4	4	4	2	3	2	1	3.17	1.27		
	UISKEI	1	5	5	5	5	5	5	4	3	5	5	5	5	4.75	0.62	4.58	0.65
		2	5	4	4	5	4	4	5	5	5	3	5	4	4.42	0.67		
		3	5	5	5	5	5	5	4	5	4	4	5	5	4.75	0.45		
		4	5	5	5	5	4	3	4	5	4	3	5	5	4.42	0.79		
		5	2	3	4	4	4	5	3	4	3	3	5	5	3.75	0.97	4.14	0.81
		6	5	4	5	4	4	5	3	4	4	3	5	5	4.25	0.75		
		7	4	3	5	3	4	5	4	3	4	3	5	4	3.92	0.79		
		8	4	4	5	4	3	4	4	3	5	3	5	4	4.00	0.74		
		9	5	4	5	5	4	4	3	5	4	3	5	5	4.33	0.78		
		10	5	5	5	5	5	5	4	4	3	4	5	5	4.58	0.67		

Table 13: Third cycle questionnaire answers and statistics.

			p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	Ave	Std Dev	Ave	Std Dev
3rd Cycle	Paper	1	5	5	5	5	5	5	5	3	5	5	5	5	4.83	0.58	3.67	1.36
		2	1	3	3	5	3	5	2	2	4	5	3	5	3.42	1.38		
		3	5	3	3	5	2	5	3	5	2	1	4	5	3.58	1.44		
		4	1	2	4	4	2	4	4	3	2	2	2	4	2.83	1.11		
		5	5	2	5	5	4	5	4	2	4	5	5	5	4.25	1.14	3.18	1.58
		6	1	2	3	5	2	5	2	1	5	5	2	1	2.83	1.70		
		7	4	3	5	5	3	5	4	2	5	5	1	4	3.83	1.34		
		8	4	3	4	5	3	5	2	2	5	5	1	1	3.33	1.56		
		9	2	1	4	3	1	4	3	1	3	2	1	1	2.17	1.19		
		10	5	1	4	3	2	5	3	1	1	1	5	1	2.67	1.72		
	Balsamiq	1	5	5	4	5	5	3	5	5	5	4	5	5	4.67	0.65	3.88	1.28
		2	4	5	2	5	3	4	1	3	3	4	5	2	3.42	1.31		
		3	5	5	5	5	1	5	5	3	2	4	5	2	3.92	1.51		
		4	2	4	3	5	1	4	5	3	3	4	5	3	3.50	1.24		
		5	5	4	5	5	4	4	4	2	3	5	4	5	4.17	0.94	2.78	1.49
		6	1	4	1	5	2	3	1	2	1	2	2	1	2.08	1.31		
		7	4	5	2	5	3	5	4	3	3	4	1	4	3.58	1.24		
		8	4	4	2	4	3	5	1	3	2	3	1	1	2.75	1.36		
		9	1	2	1	4	2	3	2	1	1	1	1	1	1.67	0.98		
		10	1	4	1	5	1	3	3	3	1	5	1	1	2.42	1.62		
	UISKEI	1	5	5	5	5	5	5	5	3	5	5	5	5	4.83	0.58	4.73	0.54
		2	5	4	4	5	4	5	4	5	5	5	5	5	4.67	0.49		
		3	5	5	5	5	5	5	4	5	3	5	5	5	4.75	0.62		
		4	5	5	5	5	5	4	4	5	5	4	5	4	4.67	0.49		
		5	4	4	5	4	5	5	4	4	5	4	5	5	4.50	0.52	4.38	0.68
		6	5	4	4	4	5	5	5	4	4	3	5	5	4.42	0.67		
		7	5	4	4	3	5	5	4	4	4	3	5	4	4.17	0.72		
		8	5	4	5	3	5	5	4	3	4	3	5	4	4.17	0.83		
		9	5	4	4	5	5	4	3	5	4	4	5	5	4.42	0.67		
		10	5	5	5	5	5	5	4	4	3	4	5	5	4.58	0.67		

Lastly, the following table shows the answers to the interview questions (presented in section 8.1), with the same letter code previously used (p=Paper, b=Balsamiq, u=UISKEI). The last columns show the number of participants who chose the corresponding tool in that question.

Table 14: Interview answers.

		p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	P	B	U
Interview	1	B	U	P	B	U	P	B	P	U	U	B	B	3	5	4
	2	B	B	B	U	U	B	B	B	B	B	B	B	0	10	2
	3	U	U	U	U	U	U	P	U	U	U	U	U	1	0	11