Bachelor Informatica

Universiteit van Amsterdam

CLIsis: An Interface for Visually Impaired Users of Apache Isis Applications

Sander Ginn

May 17, 2016

Supervisor(s): Maarten Marx (UvA), Dan Haywood (Apache)

Signed: Maarten Marx (UvA)

Abstract

This will be the abstract.

Contents

CHAPTER 1

Introduction

Every application that offers a method of interaction with a human being requires some form of user interface to enable this interaction. But what exactly characterises a user interface? The Oxford English Dictionary provides the following definition for "interface"[1]:

> **in·ter·face**, *n.*: A means or place of interaction between two systems, organizations, etc.; a meeting-point or common ground between two parties, systems, or disciplines; also, interaction, liaison, dialogue.

Thus, in the context of an application, a user interface is a means of interaction between the user and system based on common ground between these two entities. The common ground is a layer on top of the system which presents information from the system in a comprehensible manner to the user and translates user input to system-interpretable commands.

This is still a very abstract definition of what a user interface does. It does not specify how system information is presented, how the user provides input, what permissions and restrictions the interface must adhere to, et cetera. Nielsen[2] describes an overview of user interfaces throughout history which will provide some practical examples.

Batch systems were one of the first user interfaces to be introduced. To interact with the system, a user submitted all of the system commands in one batch and had to wait for all of them to be processed before being presented with the results. It was a tedious and inconvenient way of interacting with systems, as erroneous commands would prevent the batch from successfully completing. Early batch systems also often required special operators to program the commands. Contemporary user interfaces generally still offer batch operation in some form or another, as its main advantage is that it can be used to perform menial tasks without continuous human supervision.

As the use of computers became more widespread, the *line-oriented interface* was adopted as a method of interaction. The name is derived from the characteristic that the user interacted with a system on a single line; once submitted, the command could not be modified anymore. This characteristic enforces a use pattern where a dialog between the system and user takes place. The user answers questions posed by the system and issues commands with parameters. As long as the information is well structured this method of interaction still has valid use cases in present times, as it effectively delimits functionality, preventing novice users from running into issues.

The successor of line-oriented interfaces is the *full-screen interface.* This interface aims to exploit a greater deal of the screen real estate by enabling the user to interact with more elements than just one input line. The full-screen interface also introduced the concept of menus, where functionality available to the user is listed in a hierarchical structure. Menu design has been researched extensively[3, 4, 5] to determine the optimal balance of depth and breadth. Depth decreases complexity of menus while increasing ease of navigation; the contrary applies to breadth. Full-screen interfaces are still used present-day (e.g. electronic forms), albeit in adapted forms such as pop-up windows.

Finally, the user interface type which is currently the most widespread, is the *graphical user interface* (GUI). The vast majority of modern applications offer a graphical user interface as their primary method of interaction. A key characteristic of GUIs is that interaction is offered through direct manipulation. Rather than issuing a command which exactly specifies those parameters of what has to happen, the user gets continuous feedback of their input commands. An example is resizing a window: if a user were to do this by a command, the dimensions are provided as parameters and the window resizes. In a GUI, the user can dynamically resize the window with the mouse. The obvious advantage of a GUI is that it can take advantage of the human sight to represent data in a more intuitive way. Research has shown that when the interface adheres to some fundamental principles, a user's cognitive performance increases with a GUI. An example of one of these principles is Miller's Law, proposed in his seminal research on information processing[6]. However, Nielsen also addresses the issue that motivates this research: a GUI sidelines the visually impaired from using an application.

Enabling the visually impaired to use applications despite a lack of or very bad vision has been an ongoing effort since GUIs were widely accepted as the de facto standard[7]. However, it has proven to be difficult to design a GUI that is also accessible without visual cues. A big issue is that standardisation has proven ineffective: even though the United States has passed laws that enforce accessibility of websites[8] and the World Wide Web Consortium has developed the Web Content Accessibility Guidelines[9], research has shown that these guidelines are very often ignored and thus websites are rendered inaccessible for accessibility tools such as screenreaders[10].

This research aims to provide an alternative interface for visually impaired users for applications developed with the Apache Isis[11] framework. This interface will not serve as a replacement for the existing interface nor blend in with it. The interface is derived from the metamodel of the framework and thus content independent. This means that the interface is readily available for any Isis application and no alterations are necessary to add it to an existing application.

1.1 Research questions

The central research question in this thesis is:

> *Can a content independent graphical user interface be adapted to a non-visual interface so that visually impaired users are able to interact with the application?*

For the non-visual interface to be useful it is imperative that it offers the same functionality as the graphical user interface. Therefore, we will also answer the following subquestion:

> *When implementing a non-visual interface, can the integrity of the domain model be maintained while providing an effective and simple method of interaction?*

Finally, it is desired that the non-visual interface does not severely impact user performance, as this would imply that it is not a legitimate alternative to the existing interface. Thus, a second subquestion will be answered:

> *Compared to the standard interface, how is performance affected when the user employs the new interface?*

1.2 Overview of thesis

A short overview of what this thesis will describe is given.

CHAPTER 2

Theoretical background

User interface design is a thoroughly studied discipline with strong roots in psychology. In the 1980s GUI development exploded due to better hardware[12]. This meant that traditional user interfaces had to be redesigned to accommodate to the graphical features of the modern computer. In section 2.2 we will provide a brief history on how this was achieved and what sort of issues arose when migrating a user interface. Furthermore, section 2.1 will describe what Apache Isis entails.

2.1 Apache Isis

As a software engineer, picking a framework for developing web applications can be a tedious process. There are dozens of frameworks for Java alone, with the oldest and most adopted one being Spring[13]. The vast majority of these frameworks are based on the *model-view-controller* (MVC) pattern, where the view displays information to the user, the controller processes interaction between the user and the view, and the model contains the information and logic that manipulates this information[14]. The relations between the components are depicted in figure 2.1.

While the MVC pattern itself has a lot of advantages, it has received criticism in the context of web development. The concept of separating business logic from presentation logic is often not adhered to in web applications, resulting in controllers that are contaminated with logic that should live in the model[15].

This is where Apache Isis differs from more classic MVC-based frameworks.

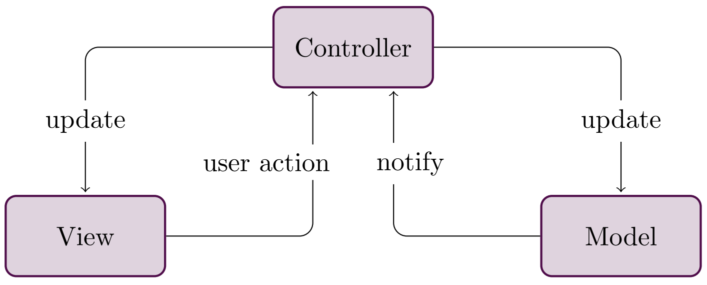

Figure 2.1: Interaction between the components of an MVC application

2.1.1 Domain Driven Design

As Apache Isis is a framework based on Domain Driven Design, I will briefly describe the core concepts of DDD.

2.1.2 Naked Objects pattern

Apache Isis implements the Naked Objects Pattern, which is related to DDD.

2.2 User interface migration in history

This section will provide some context around the issues that arise when implementing a new interface.

2.2.1 Issues and caveats

CHAPTER 3

Methods

This chapter will describe the employed methods.

3.1 User interface migration

This section will describe what method I have applied to perform the user interface migration.

3.1.1 Detection

Step one of the migration method.

3.1.2 Representation

Step two of the migration method.

Specifications

The set of specifications that are derived from section 3.1.1 are presented.

3.1.3 Transformation

Step three of the migration method.

3.2 Experiments

A description of the methods used to conduct the experiments.

3.2.1 GOMS

The Goals, Operators, Methods, Selection rules method is explained.

3.2.2 Time trial

A custom time trial method is described.

CHAPTER 4

Implementation

This chapter describes the practical side of the project.

4.1 Frameworks and tools

This section expands on what available software tools I have used to implement the new interface.

4.1.1 REST API

This subsection will describe how Apache Isis' REST API works and how it is incorporated in the new interface.

4.1.2 AngularJS

Here I will explain the choice for AngularJS.

4.1.3 Web Speech API

This subsection will describe how the Web Speech API is used.

4.2 Functionality

This section will expand on the individual components that process the information from the REST API.

4.2.1 Main module

This subsection describes the main app module.

4.2.2 Controllers

This subsection describes each controller's functionality.

4.2.3 Views

This subsection describes the purpose of each view.

CHAPTER 5

Evaluation

This chapter describes the experiment results and answers the research questions.

5.1 Specification requirements

In this section, I will motivate which specifications have been met in the implementation. With this information I will be able to answer the first subquestion.

5.2 Experiments

5.2.1 GOMS

Here, the results from applying GOMS will be evaluated.

5.2.2 Time trial

Here, the results from the time trial experiment will be evaluated.

5.2.3 Performance difference

This subsection will evaluate the experiment results and answer the second subquestion.

CHAPTER 6

Conclusion

Here, I will answer the research question and make any other conclusions.

6.1 Future work

Any remaining future work can be discussed here.

6.2 Acknowledgements

Bibliography

[1] "interface, n. : Oxford english dictionary." `http://www.oed.com/view/Entry/97747?result=1&rskey=X7p96Q&`. (Accessed on 05/16/2016).

[2] J. Nielsen, *Usability engineering.* Elsevier, 1994.

[3] K. R. Paap and R. J. Roske-Hofstrand, "The optimal number of menu options per panel," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 28, no. 4, pp. 377–385, 1986.

[4] T. K. Landauer and D. Nachbar, "Selection from alphabetic and numeric menu trees using a touch screen: breadth, depth, and width," *ACM SIGCHI Bulletin*, vol. 16, no. 4, pp. 73–78, 1985.

[5] D. L. Fisher, E. J. Yungkurth, and S. M. Moss, "Optimal menu hierarchy design: syntax and semantics," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 32, no. 6, pp. 665–683, 1990.

[6] G. A. Miller, "The magical number seven, plus or minus two: some limits on our capacity for processing information.," *Psychological review*, vol. 63, no. 2, p. 81, 1956.

[7] L. H. Boyd *et al.*, "The graphical user interface crisis: Danger and opportunity.," 1990.

[8] "Section 508 of the rehabilitation act." `http://www.section508.gov/section-508-of-the-rehabilitation-act`. (Accessed on 05/16/2016).

[9] "Web content accessibility guidelines (wcag) 2.0." `https://www.w3.org/TR/WCAG20/`. (Accessed on 05/16/2016).

[10] S. Leuthold, J. A. Bargas-Avila, and K. Opwis, "Beyond web content accessibility guidelines: Design of enhanced text user interfaces for blind internet users," *International Journal of Human-Computer Studies*, vol. 66, no. 4, pp. 257–270, 2008.

[11] "Apache isis." `http://isis.apache.org/`. (Accessed on 05/16/2016).

[12] B. A. Myers, "A brief history of human-computer interaction technology," *interactions*, vol. 5, no. 2, pp. 44–54, 1998.

[13] "Spring." `https://spring.io/`. (Accessed on 05/16/2016).

[14] A. Leff and J. T. Rayfield, "Web-application development using the model/view/controller design pattern," in *Enterprise Distributed Object Computing Conference, 2001. EDOC'01. Proceedings. Fifth IEEE International*, pp. 118–127, IEEE, 2001.

[15] "Fulfilling the promise of mvc." `https://www.infoq.com/articles/Nacked-MVC`. (Accessed on 05/17/2016).

Acronyms

GUI	graphical user interface.
MVC	model-view-controller.

APPENDIX A

Application manual

A.1 Installation

This appendix provides instructions on running the demo application.

A.2 User manual

A concise manual to use the interface.

APPENDIX B

Time trial test class

Here I will describe how the time trial has been implemented in Java.