Camilla Lamberg & Anders Brundin

Evaluating the future development options for Ozlab

Informatik

C-uppsats

Datum/Term: Vt-11
Handledare: John Sören Pettersson
Examinator: Remigijus Gustas
Abstract

The Ozlab user-test facility at Karlstad University needs to be reprogrammed. This paper will be used as a decision support document for the redesigning. Ozlab is currently depending on a discontinued software, the learning curve is too steep for new users and there are a number of flaws in the system that force the user to take detours to complete some simple tasks. Ozlab uses the Wizard-of-Oz technique. The Wizard-of-Oz technique is a method used for conducting user tests on prototypes with no functionality; instead a “wizard” controlling the test person’s computer from the next room provides the interaction. There are no general Wizard-supporting tools in the world except for Ozlab. When developing the solution for Ozlab it is important to keep this unique concept, but make the Ozlab system fit for the future and long term sustainable.

For this paper, interviews with different kinds of users were conducted to map what is most important for them in a future Ozlab. Moreover, reliance on other programs and file formats than the present ones is also discussed. These other programs and file formats are evaluated on how well they support the optimal workflow for creating and testing a prototype in Ozlab. Recommendations are made with the conclusion that an XML-based solution is the most appropriate option, because the biggest concerns are to make Ozlab as independent from software and file formats as possible, and to make sure that maintenance is easy. Other solutions discussed are based on the Adobe Photoshop software and the HTML5 format.
Credits

In order to conduct this paper, there are several people who have been very helpful.

The Authors would like to send special thanks to:

**John Sören Pettersson**
Our tutor for this paper and the originator of Ozlab.

**Lennart Molin**
Interviewee and researching Ozlab user.

**Jenny Nilsson**
Interviewee and commercial Ozlab user.

**Julio Angulo**
Interviewee and Ozlab assistant.

**Maria Lindström**
Interviewee and Ozlab assistant.

**Odd Fredriksson**
Teacher in research method.

**Björn Brundin**
Proof-reader.
Distribution of work

All chapters in this paper were written collectively with three exceptions.

Chapter 4 was written by Lamberg alone, as well as chapter 7.2.

Chapter 7.1 was written by Brundin alone.

All editing have been collectively, including the chapters we wrote separately.

For the interviews conducted for this paper, we simply took turns interviewing and taking notes every other time.

During the actual writing we almost exclusively had one author typing at a time while discussing with the other one, and then we just took turns.

Considering literature most of it was gathered and analyzed collectively.

All original figures in this paper have been produced collectively.
Table of Contents

Abstract ........................................................................................................................................... 1
Credits ............................................................................................................................................. 3
Distribution of work .......................................................................................................................... 4
  All original figures in this paper have been produced collectively .............................................. 4
Figures ........................................................................................................................................... 7
Tables ............................................................................................................................................ 7
1 Introduction ................................................................................................................................ 1
  1.1 Purpose ................................................................................................................................ 1
  1.2 The background of Ozlab ....................................................................................................... 1
  1.3 Ozlab Users ........................................................................................................................... 2
  1.4 Chapter Overview .................................................................................................................... 3
2 Prototyping non-programs, interactive prototypes ................................................................... 4
  2.1 Defining the Ozlab Prototype ................................................................................................. 4
  2.2 Usability .................................................................................................................................. 5
  2.3 User/Usability Tests ................................................................................................................ 6
  2.4 The Wizard-of-Oz technique .................................................................................................. 7
  2.5 The Naive Ozlab Wizards ....................................................................................................... 8
3 Method ....................................................................................................................................... 9
  3.1 Interviews ............................................................................................................................... 9
  3.2 Focus Group ........................................................................................................................... 10
  3.3 Analysis of Workflows .......................................................................................................... 10
  3.4 Analysis of File formats ......................................................................................................... 10
4 Description of current system ................................................................................................... 12
  4.1 Ozlab programs ..................................................................................................................... 12
    4.1.1 Ozlab Setup 1.4 ............................................................................................................... 12
    4.1.2 Ozlab FileUpdater 1.4.2 ............................................................................................... 12
    4.1.3 Ozlab Testrunner ........................................................................................................... 13
  4.2 Macromedia Director MX ...................................................................................................... 16
    4.2.1 Introduction to Macromedia Director ............................................................................. 16
    4.2.2 Creating interaction-shells ............................................................................................. 23
  4.3 The multiuser server computer .............................................................................................. 24
5 User groups requirements .......................................................................................................... 25
Figures
Fig 1.1 – Wizard of Oz illustration ..........................................................2
Fig 4.1 – The Ozlab system window ..........................................................12
Fig 4.2 – Ozlab Shell example .................................................................13
Fig 4.3 – Change-shell window ...............................................................14
Fig 4.4 – Marker navigation window ......................................................14
Fig 4.5 – Activity list window .................................................................15
Fig 4.6 – Control panel window .............................................................15
Fig 4.7 – Director window ....................................................................17
Fig 4.8 – Stage window ..................................................................18
Fig 4.9 – Cast window ..................................................................19
Fig 4.10 – Score window .................................................................19
Fig 4.11 – Property inspector window ................................................20
Fig 4.12 – Tool palette ..................................................................21
Fig 4.13 – Paint window ..................................................................22
Fig 4.14 – Text window ..................................................................23
Fig 6.1 – Activities in the Ozlab prototype process ..............................33
Fig 7.1 – Photoshop Solution ...............................................................34
Fig 7.2 – Photoshop Workflow ............................................................35
Fig 7.3 – HTML5 Solution .................................................................36
Fig 7.4 – HTML5 Workflow ...............................................................37
Fig 7.5 – XML Solution .................................................................38
Fig 7.6 – XML Workflow ...............................................................39

Tables
Table 8.1 – Comparison Table ...............................................................42
1 Introduction

1.1 Purpose

This paper discusses the future development of the Ozlab system and different options for improving and modernizing the Ozlab system at Karlstad University. This paper will be used as a decision support document when Ozlab is redeveloped, while other people will contribute other reports.

The current solution is dependent on Macromedia Director MX and its scripting language LINGO which is outdated and needs to be replaced with a more long-term solution. The current solution requires that test moderators master Macromedia Director MX. Since Macromedia Director MX was discontinued in 2005 the knowledge levels of test moderators using Ozlab varies significantly from older test moderators to newer who never had a reason to learn Director and its programming language. Our focus in this paper will be to investigate what different software options are available and possible to incorporate into Ozlab. The software will have to be a solution for both the current situation as well as in the long run, by this we mean that future releases will have to be backwards compatible as to not make our solution obsolete in a few years. Another important aspect of our investigation is how all this will meet the current functionality standards of Ozlab.

1.2 The background of Ozlab

The Ozlab system is a test-environment at Karlstad University where user tests are conducted on prototypes according to the Wizard-of-Oz technique. This technique is based on the fact that you fool the test person to think that he or she is testing a partially or fully working product while all the interaction is really provided by a test moderator sitting in the next room, as shown in figure 1.1.

The technique is mostly used for testing user interfaces for software and websites and got its name at Johns Hopkins University during the 1980’s. It is named after the famous book The Wonderful Wizard of Oz from 1900 (Baum 1900). For more about the Wizard-of-Oz technique see section 2.4 - the Wizard of Oz technique.

The Ozlab system itself was developed at Karlstad University and has now reached a point where it needs to be modernized and updated. It consists of the Ozlab Testrunner by which the actual interaction tests are conducted, a software to build the interaction-shells as the prototypes are referred to in Ozlab and the Ozlab FileUpdater. For more about the different software in Ozlab see chapter 4 - Current system description. The current software that is used when building the interaction-shells is Macromedia Director MX, which makes the Testrunner totally dependent on this software and its script language LINGO. It is of great importance that this dependency is eliminated since Macromedia director MX was
discontinued in 2005. A new solution is therefore needed, either by accommodating the Testrunner to a new software that will not be discontinued for a long period of time, or accommodate it to a certain file format that can be used by a number of software.

The unique thing about Ozlab is that a test moderator is able to change the appearance and behaviour of a prototype between or even during test sessions. Larsson & Molin (2005) states that when test persons usually make suggestions after or during tests, it often concerns overall design issues such as workflow orders or where functions should be added or removed. In normal system development with programmed prototypes, alterations such as these would be extremely time-consuming.

“The possibility of modifying a prototype according to the users’ suggestions helps giving them a feeling of control. They are able to see their suggestions take shape and this could notably motivate them and possibly strengthen the collaboration between the users and the developers.” (Larsson & Molin, 2005)

[Fig 1.1] The wizard on the right side (test leader) controls the test person’s computer (on the left) through scenarios in an Ozlab prototype. Illustration by Anders Karlsson

1.3 Ozlab Users

Primary users of the Ozlab system in its current state are employees and students at Karlstad University. In the future this will likely include people outside of the university as secondary users. These users form the target group which is currently made up of people both with and without knowledge in usability testing and interaction design. The University would like to see that the knowledge of Ozlab spreads both within the university and to interested enterprises.
In this essay we discuss five different user groups and their requirements on the Ozlab system. These are the user groups we have defined:

- **Teaching users**
  Everyone who use Ozlab as a tool when teaching usability, prototyping and user testing. Teaching users can be branched out into two smaller user groups, the Ozlab Assistants and the researching users
  
  o **Ozlab assistants**
    An Ozlab assistant is employed by the University and is responsible for the maintenance and also assists the teachers when Ozlab is used for educating students.
  
  o **Researching users**
    Every usability expert who use Ozlab as a tool during research, or researches Ozlab.

- **Student users**
  All students who in any way use Ozlab during their education.

- **Commercial users**
  People who use Ozlab outside the University.

- **Naïve users**
  Users who have no prior knowledge about Ozlab, usability or user testing. All users that cannot be identified with any of the previous user groups, we refer to as naïve users, with one exception for anyone who is an expert in usability, user testing, prototyping or system development. Students can be thought of as naïve users, but since the students receive frequent tutoring in the handling of Ozlab, they are not included in the naïve user group.

### 1.4 Chapter Overview

2. **Prototyping non-programmed interactive prototypes** – Describes and briefly discusses the literature that this paper is based upon.

3. **Method** – The methodology used for the analysis in the paper

4. **Description of current system** – Describes How Ozlab was working during the time this paper was written

5. **User groups requirements** – Presentation of, and analysis on the collected requirements

6. **The Optimal Workflow** – Describes the workflow that needs to be supported by the future Ozlab.
7. The Possible Solutions – Presentation of and evaluation of the different solution options.

8. Recommendations – Comparison and evaluations between the solutions as well as our recommendations for the future development of Ozlab.

2 Prototyping non-programmed interactive prototypes

There are no books concerning Ozlab. Because of this, all literature gathered concerning this is from published reports or from conference material.

When it comes to literature concerning usability and user tests, we have referenced and discussed well known and proven theories, even if they are of age. However, we have based much of our analysis later in this paper on these theories. So it is truly relevant to show the basic principles a reader needs to understand.

2.1 Defining the Ozlab Prototype

Prototypes are categorized by stating its fidelity. Fidelity is a measurement of how close to the finished product the prototype works and is built (Farnum 2002). Lo-fi (low-fidelity) prototypes are rough sketches which are a very basic structure with little or no interactive functionality. It is mostly used in the early stages of the development to get a greater perspective of the design concept. The opposite is the hi-fi (high-fidelity) prototypes which are as close to the finished product as possible. It can be a lifelike simulation of the system as it might look once it is released. The hi-fi prototype is used as marketing tools or training tools for its users and also to confirm that usability problems truly have been removed (Engelberg & Seffah 2002). Another way to define lo- and hi-fi prototypes according to Nilsson and Siponen (2005) is depending on the way test persons interact with the prototype. If it is obvious that the prototype is handled by another person, it is a lo-fi prototype, as with paper prototypes it is quite obvious when someone changes the paper in front of the test person. On the other hand, the hi-fi prototype is when all the interactivity with the system is automatic.

These two levels of fidelity are the commonly recognized in literature today. There are discussions of adding a middle level of mid-fi (Middle-fidelity) prototyping since the definitions of where lo- and hi-fi start and end are very loosely defined (Engelberg & Seffah 2002). This leads to a gray zone in between lo- and hi-fidelity where the exact location of the border is hard to define. However, even with a clear border it would not solve the classification of an Ozlab prototype:

One would expect that placing the Ozlab prototype (interaction-shells) in the correct fidelity category is accomplished easily. On the surface, an interaction-shell appears to be of high fidelity if the measure of fidelity assumed to be based
on the user’s expectations of an interactive system. Conversely, from the perspective of the developer, the very same prototype would seem to be of low fidelity, because it is incomplete as an automatic system. (Nilsson & Siponen 2005)

Thus, as Nilsson and Siponen (2005) mentions there are no clear way to define what kind of fidelity the Ozlab prototype is. It is a hi-fi prototype if you look from the test person’s point of view, but for the wizard that controls the test, it is a lo-fi prototype.

The Ozlab prototype is a mixture of the three above mentioned levels of fidelity. It has the option to just be scanned sketches shown to the test person, to a finished design with behaviours appearing to be fully functional. In the lo-fi scenarios the Ozlab system loses some of its appeal, since it will be quite obvious that there is a wizard controlling what you see. On the other hand, a hi-fi prototype with a lot of functionality and behaviours with no limits will be too much for a wizard to handle smoothly (Pettersson 2001). A mid-fi prototype that shows some functionality and design will be optimal to test in Ozlab since this is within the boundaries currently set up by the Ozlab system.

2.2 Usability

Usability is best explained with the definition made by the International Organization for Standardization:

"Extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use." (ISO-9241-11, 1998)

Ozlab is mainly used for evaluating the usability of a system through a prototype.

When planning, conducting and evaluating a user test session, the 10 principles developed by Nielsen (2001) presented below are often used to define what should be tested in test sessions. Nielsen originally made these principles for expert evaluations of systems. They work well to categorize findings when writing your test report. These principles are not used as a foundation for evaluation in this paper, but they are used where they are relevant to further the discussion.

Visibility of system status
The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

Match between system and the real world
The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
User control and freedom
Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

Consistency and standards
Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

Error prevention
Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

Recognition rather than recall
Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

Flexibility and efficiency of use
Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

Aesthetic and minimalist design
Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

Help users recognize, diagnose, and recover from errors
Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

Help and documentation
Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

2.3 User/Usability Tests
Testing a system with potential end users is one of the most effective ways to evaluate it. The purpose of user tests is to find flaws and weaknesses in the system Nielsen (2001). User tests conducted in the Ozlab system are mainly focused on usability rather than functionality, since
Ozlab itself is based on the Wizard-of-Oz technique, which means that all system functionality is faked (see section 2.4 Wizard-of-Oz technique).

2.4 The Wizard-of-Oz technique

The Ozlab system is based on the Wizard-of-Oz technique. The technique is basically used to test prototypes and systems that will be seen by the test person as autonomous, while in fact they are simulated by a test moderator (the wizard). Like its novel namesake by Baum\(^1\), its about manipulating the test person into thinking the program they are testing is autonomous, while the wizard sits in another room and watches the test person through a camera or a one-way mirror and simulating the interactivity with a computer. As the Wizard of Oz says in the 1939 film based on Baum’s novel:

"Pay no attention to that man behind the curtain." (The Wizard of Oz. 1939\(^2\))

The technique is used for testing a prototype that the test person thinks is partially or fully working without the interaction designer having to program the functionality. The benefit of this technique is that time and money is saved when the amount of re-coding is minimized.

The name of the technique has its origin in the dissertation of John. F. Kelley concerning the Natural Language Processing system “Calendar Access Language” (CAL) at The Johns Hopkins University around 1980 while the technique itself had been used previously at Johns Hopkins University. Originally Kelley referred to the technique as the OZ paradigm (Kelley 1984).

The development process for CAL [...] Central to the methodology is an experimental simulation which I call the OZ paradigm, in which experimental participants are given the impression that they are interacting with a program that understands English as well as another human would. (Kelley 1984)

Since the beginning of the 1980's the Wizard-of-Oz technique has been used in a variety of ways, mainly in usability testing of user interface design as well as prototypes and the development of these.

Kelley (and others) used it to prototype natural language processing (NLP) without any graphical components. When graphical user interfaces (GUI) were introduced in the 1990’s the Wizard-of-Oz technique had to evolve. Ozlab was created to easily add images and texts to a prototype and thus making it a more modern tool to use with graphical user interfaces.

---

\(^1\) Baum, L.F. (1900). *The wonderful Wizard of Oz*. Chicago, George M. Hill Company.

2.5 The Naïve Ozlab Wizards

In this section we make an account of the Ozlab developers’ view on naïve wizards and how they can contribute during user testing in Ozlab. Molin and Pettersson state that non professional designers are able to use user testing programs, and that there could be a benefit in involving non professional designers in Ozlab user testing.

Claiming that Ozlab is an easy-to-use, combined usability testlab and workstation for explorative interaction with users, entails a further claim—or at least a goal—that such a laboratory can be used by laymen to define the design of future systems. Simple multimedia tools for school use [...] show that non-programmers can take on a role of designers-and-implementers, at least when the systems built are fairly simple. Moreover, non-programmers are professionals in their own field of work and could have some advantages over the professional designers. (Molin and Pettersson. 2005)

During the autumn of 2001, Ozlab was used in an experiment to test multimedia user interfaces for children with Down’s syndrome. During these tests three members of the Disability and Language research group at Karlstad University participated as wizards (Pettersson 2002). There are benefits when involving naïve wizards with a special connection to the purpose of the system and its end users. Since the system is within their area of expertise they can contribute with more content knowledge and enthusiasm than a wizard knowledgeable in interaction design.

The enthusiasm of our wizards never seriously failed when a test session had to be delayed an hour or so. There was obviously enough inspiration and thrills from interacting with the clients in this deceptive way to keep the wizards interested in and even excited about conducting their experiments. (Ibid.)

Pettersson (2002) mentions that the naïve wizards in this particular experiment needed the company of a programmer to both create interaction-shells and conduct the tests properly.
3. Method

In this chapter, we present the methods used in this paper and how they best can be used. The methods we used were interviews and focus groups. We also analyzed file formats suitable for Ozlab prototyping as well as the workflows.

The research for this paper is based upon qualitative and explorable knowledge gathered from interviews and literature. The future options that are suitable for Ozlab need to be developed and based upon extensive and complete facts. To gather the knowledge needed to make such a paper, Patel & Davidson (2003) suggests interviews, focus groups and existing academic reports.

3.1 Interviews

For this paper several interviews were conducted. The four people interviewed represent all the different user groups defined in section 1.3 except the student users. The interviewees were chosen from a small target group of people with close ties to Ozlab. The original plan was to interview five people, but since contact was lost with one of them who were on maternity leave, and there was no time to wait for further contact, only four interviews were conducted. A few of them were involved in the actual development of Ozlab, others are teachers at Karlstad University and who are currently teaching students about Prototyping, usability and the actual Wizard-of-Oz technique. Our interviewees are people that have educated students in Ozlab for an extended period of time, so what was needed from our interviewees were their thoughts on how Ozlab might evolve and why this is needed to further the progress of prototype testing in educational situations. Ozlab in its current state is still relatively unknown outside of Karlstad University and is therefore mostly used at Karlstad University. The goal was to gather input from several different types of current and future Ozlab users.

The interviews emanated from the purpose with this essay, and the questions were based upon that purpose as well (Kvale 1997). A personal approach was needed for our interviews, and all the interviews were conducted face to face. Before an interview began the interviewer should inform the interviewee if the interview will be confidential or not (Patel & Davidson, 2003). Confidentiality couldn’t be promised to the interviewees since they have to interact a lot with each other and it would be obvious whose opinions are whose. The interviewees were not asked to predict the future of Ozlab, but to talk about how they would, ideally like to use Ozlab in the future. As Kvale (1997) stated, the purpose of qualitative interviews are to understand real life through the eyes of the interviewee.

Questions for the interviews (see Appendix A), as well as follow-up questions were prepared and studied so that all the relevant information the interviewee possessed was collected by the interviewers. These questions were used as guidelines during the interviews, therefore the
analysis in chapter 5 is not structured based upon the questions asked, but on the important opinions raised. The questions prepared were used to keep the interviewees “on track”. This means that it was of importance to let the interviewees know that their contribution of knowledge was essential for our investigation.

To get an overview of the most frequently discussed issues from the interviews we wrote them down on post-it notes, put them on a white board and tried to categorize them into topics. These topics were later used for structuring the discussion in chapter 5 - User groups requirements.

3.2 Focus Group

Our initial thought was to gather a focus group out of first and second year interaction design-students, to capture their thoughts and requirements on the future Ozlab.

Unfortunately, we did not find enough students interested in helping us out with this, so instead we decided to go through some reports that were made by interaction design-students in the fall of 2010, when they were asked to describe their thoughts concerning a development of Ozlab. These were third year students, but we still consider them as representatives for the younger and more naïve student users, since most of them had not used Ozlab in their education for almost two years when the reports were written. Also, these reports actually included interviews with first and second year students. We read these reports and try to sum up the main issues brought up by the students, and found that most of them were related to the test leader GUI and the actual facilities of Ozlab. We discuss these issues further in section 5.7.

3.3 Analysis of Workflows

The material we have gathered concerning Ozlab, including the interviews as well as our own experiences and the Ozlab User guide shows that there are certain activities that cannot be disregarded in the process of making and maintaining an Ozlab prototype. This information is necessary to evaluate different solutions for a new implementation of the Ozlab system. Therefore we map these activities in a model, which is shown and described in chapter 6. The solutions in chapter 7 are evaluated on how well they support this workflow.

3.4 Analysis of File formats

The file formats discussed in this paper are the Photoshop Format PSD, The HTML5 format and XML. Adobe Flash has previously been discussed as a solution for Ozlab by its users. A lot of the functionality available in Adobe Flash could be covered by HTML5 and since HTML5 is software independent and less likely to be outdated we find this to be a better option than Adobe Flash. The PSD file format has been discussed by a previous Ozlab assistant.
PSD and HTML5 was brought up to discussion because two essays discussing the possibilities of incorporating these formats into Ozlab were written by the present authors as course reports for a course spring 2011 (see Brundin 2011 and Lamberg 2011 in the Reference section). This made the two formats the starting point of this paper. During the writing of this paper, a third option emerged, the XML solution. This solution was conceived during the analysis of the other two solutions as well as the interviews.

The solutions based upon different file formats are evaluated in table 8.1 to see how well they meet the preferred workflow in chapter 6, as well as the requirements of the user groups. These requirements are taken from the topics discussed by different users in chapter 5 - User groups requirements.

From the evaluation a recommendation is made in section 8.2 – Recommendations.
4 Description of current system

Most of the information is directly translated from the User guide (Siponen et al. 2002). The images in this chapter are screenshots from the English version of Ozlab Testrunner.

4.1 Ozlab programs

4.1.1 Ozlab Setup 1.4

The Ozlab Setup is used for making the settings that is necessary for the Ozlab Testrunner to work properly during a test session. The settings that have to be made are:
The multiuser Server computers IP-number
The path to the Ozlab Testrunner root folder on the TL-computer.
The path to the Ozlab Testrunner root folder on the TP-computer.

These settings are saved in the sync.ini files which are fully explained in 4.1.2

![fig 4.1] The Ozlab system window

4.1.2 Ozlab FileUpdater 1.4.2

The Ozlab File Updater is installed on the TL-computer and makes sure that the files in the Ozlab Testrunner is copied from the root folder on the TL-computer to the root folder on the TP-computer. The point is that the exact same system files and interaction-shells should be available on both the TL and TP-computers during a test session. The FileUpdater also creates two start-up files that land up in the root folders in both the TL and TP-computers. Both of
these files are saved with the ending .ini but the contents differ between the two since they contain the parameters that determines if the computer acts as a TL or a TP-computer and the Multiuser-Server-computers IP-number. The File Updater gets the information it needs from the current settings in the Ozlab Setup.

### 4.1.3 Ozlab Testrunner

The Ozlab Testrunner is the software that provides the Wizard-of-Oz functionality to the TL-computer, it is also installed on the TP-computer and runs on both of them at the same time during a test session. The content displayed however looks a bit different. On the TP's computer the graphic elements in the interaction-shell is shown and it looks like an interactive program, while on the TL's computer there are a lot more details displayed. In the centre of the TLs screen a copy of the TPs screen is displayed. On the rest of the TL's screen the functions the TL needs to conduct the test is placed out in different windows. These windows are explained below.

![fig 4.2] Ozlab Shell example

In the window called ”Change shell”(fig 4.2), all the interaction-shells that are in the same folder is listed. The TL can use this window to change to another interaction-shell during the test. The active shell is marked as in fig 4.3. By clicking on the link “Standby” next to the shell name, TL displays a “wait” screen on the TP computer. This can be used if the TL needs an extra couple of seconds to do something after starting a shell without revealing this to the TP. It displays a message to the TP to make him or her think that the program is working.
Then there is a button called "Hide wait screen" on the wizard control panel (see below) to restore TP’s screen.

![fig 4.3] Change-shell window

The "Marker navigation (Page navigation)"

This window is used by the TL to navigate between the different pages in an interaction-shell. In the .dir file these pages are called markers. So what really happens in the .dir file, when the TL navigates by using this window, is that it jumps from one place on the timeline to another. As shown in figure 4.4, in this window there are two different ways of navigating, one dropdown menu and two buttons, one to go forward and one to go backward.

![fig 4.4] marker navigation window
The "activity list"
This window shows the TL all the buttons that the TP clicks during the test session. This does however require that the buttons in the interaction-shell has been provided with the behaviour that registers the clicks in the activity list.

[fig 4.5] Activity list window

The "Control panel"
This window is a control panel with a number of buttons to help the TL moderate the test. In fig 4.6 we see the "Control panel" with all of its buttons, followed by short explanations of each button.

[fig 4.6] Control panel window

**Show/Hide wait screen:** shows the text "Wait" on the TPs screen  
**Freeze TP screen:** freezes the TP’s screen and changes the cursor into an hour glass.  
**Lock moveable:** locks all the moveable objects in the interaction-shell, this lock is marked for the TL to see.  
**Reset Objects:** Resets all the objects that have been moved or changed during the test to their original state.  
**Show activity list:** Shows the window "Activity list" on the TL’s screen.  
**Change shell:** Shows the window "change shell" on the TL’s screen.  
**Exit:** shuts down the Ozlab Testrunner on both the TL’s and TP’s computer.
Opening interaction-shells in Ozlab Testrunner

All usability tests in the Ozlab system is run through Ozlab Testrunner, which runs on both the TL computer and the TP computer. To start a test, an interaction-shell is opened in Ozlab Testrunner. Directly below is described how an interaction-shell is to be properly opened. Start the Ozlab Testrunner on the TL-computer by double-click on the icon for Ozlab Testrunner the desktop. A window that says "START TEST" appears on the TL-computers screen. Start the Ozlab Testrunner on the TP-computers by double-clicking on the icon for Ozlab Testrunner the desktop. A white background with the text "Wait" is now shown on the TPs screen. This image will be displayed on the TP-computers screen until the TL starts a test from the TL-computer. Click on "Open shell" in the dialogue box on the TL-computer’s screen. In the window that is displayed shows all the files in which there are interaction-shells. Choose the file you want to access. The interaction-shells in the files will now be displayed in the window. During a test session it is possible to use more than one interaction-shell at a time and in that case all of them are displayed in the same window. Mark the interaction-shell you wish to use for you test. Press "Open".

The interaction-shell is now opened and displayed on both the TL's and TP's computer and the test can begin.

4.2 Macromedia Director MX

4.2.1 Introduction to Macromedia Director

Macromedia Director has three main windows used for creating interaction-shells as shown in figure 4.7; these windows are:

- Stage
- Cast
- Score
Other windows used are:

Property inspector
Tool palette
Paint
Text

All of these windows are available from the Toolbox or the Window-menu in Director and are fully explained below.
The stage window (fig 4.8)

The Stage is the workspace on which you put the objects that you want in your interaction-shell. These objects are graphical elements such as images or text but could video or even audio players.

The properties, such as colour, size and resolution of the Stage, can be changed in the Property Inspector as will be explained further down. The Stage is found in the window-menu and in the Tool Palette.
The Cast Window(fig 4.9)

In the *Cast*, all of the *castmembers* are listed. A *castmember* could be an object such as text, images or sounds in the current interaction-shell. It can also be the Ozlab built-in routines that give an object certain properties and behaviours. All the objects must exist as *castmembers* before they can be used in the *Stage*. The objects can be created in Director itself or be imported into the *Cast*. A specific *castmember* can be used for creating several objects of the same sort. It is possible to have more than one *Cast* open at the same time, with different content.

The Score window(fig 4.10)
In the *Score*, the objects in the interaction-shell are shown in relation to a timeline. One row in the *Score* is called a *sprite channel* and can represent one object in the interaction-shell. The columns in the timeline are called *frames* and each represents a time unit, the stage only shows one frame. Every time an object is added to the *Stage* from the *Cast* it is automatically added as a *sprite* in one of the channels in the *Score*. Objects placed in a *sprite* channel with a low number are placed behind objects in a *sprite* channel with a higher number.

**The Property inspector window (fig 4.11)**

![Property inspector window](fig 4.11) property inspector window
The *Property Inspector* is used for modifying the properties of the *Stage* and the objects placed in it. The following list contains the most important properties that can be set in the *Property Inspector*:

- **Background-colour of the Stage**
- **Size of the Stage**
- **An object's colour**
- **An object's size**
- **An object's position on the Stage**

The *Property Inspector* is found in the window-menu and in the *Tool Palette*.

**Tool Palette**

![Tool Palette](image)

*Fig 4.12* the tool palette

The *Tool Palette* contains a number of buttons for creating objects directly in the *Stage*, these objects also becomes *castmembers* in the *Cast* and *sprites* in the *Score* automatically when created in the *Tool Palette*.

The *Tool Palette* is found in the window-menu.
Paint is the built-in window for image editing in Director. It is equipped with a complete set of drawing tools and a colour palette. Here you can create and edit an image object, and any object created in *Paint* is automatically added as a *castmember* in the *Cast*. *Paint* is accessed in the window-menu and in the *Tool Palette*. 

*Paint* is the built-in window for image editing in Director. It is equipped with a complete set of drawing tools and a colour palette. Here you can create and edit an image object, and any object created in *Paint* is automatically added as a *castmember* in the *Cast*. *Paint* is accessed in the window-menu and in the *Tool Palette*. 

![Paint Window](image)
The *Text* is used for creating text-objects in Director. To create a new text you click on the plus-icon in the *Text* and then type in whatever text you like in the text-field. Any Object created in the *Text* is automatically added as a *castmember* in the *Cast*. The *Text* is found in the window-menu and in the *Tool Palette*.

### 4.2.2 Creating interaction-shells

In order to conduct a usability test in the Ozlab system an interaction-shell needs to be created. These shells are created in Macromedia Director MX and are based on a template included in the Ozlab system. This section covers the basics of Macromedia Director MX and the how it is used in the Ozlab system. This section is based on the Ozlab User Guide from 2002.
Template functions
The Ozlab system has a Director template that is used when making the interaction-shells. This template has the necessary functionality for the interaction-shells included in its cast. Below are the names of these built in functionality along with a short explanation of what they do. The functions are programmed with their Swedish names, but have recently been translated into English as well.

**objetkFlyttbartAvTL** Makes an object moveable by the TL
**objetkFlyttbartAvTP** Makes an object moveable by the TP
**objetkFlyttbartAvTLTP** Makes an object moveable by both TL and TP
**objetkGömbartAvTL** Makes it possible for the TL to hide an object from the TP
**objetkKnapp** Creates a clickable button out of any object
**objetKomIhågPosition** Makes it possible for the TL to bring an object back to its original position by clicking the “reset objects” button in the Wizard control panel.
**objetkOsynligtFörTP** Makes an object permanently invisibly to the TP
**objetkOsynligtFörTL** Makes an object permanently invisibly to the TL
**sidmarkör** the “pagemarker” is used define to define individual scenes in the interaction-shell. (This function is made by adding code in the frame script channel in the Score-window. This needs to be placed in the same column as a marker in order to work properly.)
**spelaUppLjudFörTP** Allows the TL to control the playback of an audio-file by clicking on directobjects of the type button.
**textfältEditerbartAvTL** allows the TL to edit an object of the type text-field, can be combined with the function below
**textfältEditerbartAvTP** allows the TP to edit an object of the type text-field, can be combined with the function above.

Preparations in Director before creating an interaction-shell.
Before you can start building an interaction-shell, the template for Ozlab interaction-shells needs to be opened in Director. As described above, this template contains a particular *Cast* with built in routines used to provide the different objects with the properties especially accommodated for Ozlab.

4.3 The multiuser server computer
Multiuser Server 3.0 is the server-software used in the Ozlab network; it transmits all the communication between the other computers. When Ozlab is used on a pair of laptops, the server has to be installed on either the TL or the TP laptop.
5. User groups requirements

Immediately below are the interviewees introduced in short, and also which user group he or she represents. The important parts from the interviews are presented and analysed with one section for each topic. The topics discussed in this chapter are based on the most important and most frequently discussed issues during the interviews, with the exception of 5.7, which contains the collected student thoughts.

Julio Angulo works as an Ozlab assistant at Karlstad University. He represents the Assistant user group.

Maria Lindström works as an interaction designer, but previously worked at the University as an Ozlab assistant. Because of this she represents the commercial and assistant user groups. Since the interview was mostly from an assistant point of view, this is the user group she represents.

Lennart Molin is a lecturer at Karlstad University. He uses Ozlab as a tool in his courses in interaction design. Lennart also used Ozlab for is doctoral thesis to test participatory design process of user interfaces. He represents the researching user group.

Jenny Nilsson works as an interaction designer. She wrote her bachelor- and master theses based on Ozlab prototyping. Since then she has used Ozlab in her work as an interaction designer. Jenny represents the commercial users, because the interview with Jenny made it clear that she has more opinions from a commercial user view than a student. She is the only one that does not belong to the teaching user group.

One additional person, John Sören Pettersson, the originator of Ozlab, made statements and comments on the issues raised by the interviewees. He was not chosen as an interviewee as he is the supervising teacher for this paper.

5.1 Software independence

One of the things that all users except the commercial user agree on is that the future Ozlab should not be dependent on specific software. However, the commercial user has not clearly stated an opinion about this. The Ozlab assistants have stated that the current software dependency towards Macromedia Director MX has caused many of the issues in Ozlab today.
Both the researching user and the assistants agree: an Ozlab that is independent of other software and systems is really important. In addition, one of the assistants states that an independent file format should be used. The assistant continues with suggesting XML because it would in his opinion improve the flexibility. (Angulo)

Software independence seems to be very important in order to make the future Ozlab work in a long term perspective, because we know that depending on Macromedia Director MX have caused problems in the current Ozlab system, both regarding usability and maintenance. Ozlab should not be made dependent on any existing software if possible, in order to avoid unnecessary re-programming in the future. Obviously there is a risk in depending on existing software since there is always a possibility of it being discontinued.

5.2 Naïve Users

The naïve users are defined in the section 1.3. This user group has been mentioned in works by the Karlstad researchers: two examples were given in section 2.5. This user group was also brought up in the interviews although we made no explicit questions about it (Appendix A).

The commercial user and the Ozlab assistant have opposing opinions concerning naïve users. The commercial user thinks that Ozlab should not be made available to the naïve users, while the Ozlab assistant thinks that it should. The researching user also thinks that Ozlab should be made available to the naïve users.

"Ozlab should accommodate naïve users, but not at the expense of interaction designers not being able to use it." (Molin)

We assume that the reason that the commercial user does not want Ozlab to accommodate naïve users is that they need Ozlab to be powerful, efficient and useful in their work. We think that commercial users can accept a more complex solution for Ozlab, since they have the knowledge and prerequisites to use a tool such as Ozlab. “I do not believe in the vision that the end user can use Ozlab to create his or her dream system” (Nilsson). We believe that the Ozlab assistants use Ozlab to a further extent than any other user group. Therefore they have been exposed to several usability issues concerning Ozlab that other user groups are still unfamiliar with. The teaching user group states that part of the original idea with Ozlab was that anyone could use it.

---

3 Lennart Molin Lecturer at Karlstad University, interview 17th of March 2011
4 Julio Angulo Doctoral student Ozlab assistant Karlstad University, interview 17th of March 2011
5 Jenny Nilsson ID MSB, interview 6th of April 2011
It is noticeable that the teaching users of Ozlab think that it should be developed for naïve users. But since nearly all the naïve users they have worked with have been new students which have been extensively tutored during their usage of Ozlab, the teachers’ opinion is biased on this topic. In section 2.5 – The Naïve Ozlab Wizards, a discussion is brought forward concerning pros and cons with having naïve wizards in Ozlab testing. Even if naïve wizards can contribute with content expertise and enthusiasm, Ozlab should be easy to use for the knowledgeable users of Ozlab, not an easy-to-use program for every possible user. We do not think that the naïve users have the prerequisites to make good prototypes and tests, thus not being able to use a tool like Ozlab to its full extent. Moreover naïve users/wizards will almost constantly need the help of a professional designer for prototype designing and testing. Therefore we believe that Ozlab should be developed for the knowledgeable users that can truly utilize every aspect of the Ozlab package, including any kind of future versions of Ozlab. By knowledgeable users we mean anyone with knowledge in usability, user-testing, prototyping, systems development and people studying these subjects. If naïve users really have a reason to use Ozlab, and its unique features, it should of course be possible for them to quickly learn the basics of the system. As stated in Nielsens heuristic principle number 7. Flexibility and efficiency of use, the system should have accelerators that will speed up the interaction for the knowledgeable users.

5.3 Long-term sustainability

All users had some important opinions on what is needed to achieve long-term sustainability. Two major opinions were brought forward by the interviews: maintenance and well structured source-code.

Maintenance of the Ozlab system has never existed. It is really important, it does not matter which product is chosen. One can choose a commercial product or develop a new one, it has to be maintained. One cannot just develop a system and then say it is done. There has to be built maintenance-routines around the system, otherwise one throws away money developing a new system. (Nilsson5)

The researching user stated that, unlike the current solution, having a well structured and commented source-code will be a huge advantage concerning maintenance and future modifications (Molin3). The commercial user as well as the Ozlab assistants stated that Ozlab has not been maintained in any way (Lindström6). The commercial user say that when creating the new Ozlab there has to be well thought out plans on maintenance (Nilsson5).

Thus maintenance and quality of source code are two main issues brought up during the interviews when requirements for long-term solutions were asked for. Since avoiding maintenance can cause unnecessary problems in the future we think that it is of great importance to create clear maintenance-routines. This would not only prevent technical

6 Maria Lindström ID Make IT Simple, interview 16th of March 2011
issues, but also make the handing over easier, when replacing the staff responsible for maintenance. John Sören Pettersson, who is the creator of Ozlab, gave his opinion on what has been the problem of maintaining Ozlab codes. He points out that:

Maintenance costs have to be covered by some budget: for a commercial program it is the revenues from selling licenses, but it takes quite a substantial customer base to make this self-sustaining. In a university case, it is research projects and teaching which could provide the necessary revenues. For research, it has paid for the translation to English and sometimes also for hardware which of course is necessary to run anything at all, but it is a bit demanding to rely on research projects for continuously up-date auxiliary software such as Director and do reprogramming of the Ozlab system for every up-date in Director or Windows. For education we have sources to finance the lab, and furthermore laptops for students to borrow, etc. These are things that are not only directed towards the Ozlab system. Teaching has more continuity than external research grants, but there are other problems for continuous maintenance, which brings us to the second topic you mentioned. The present Ozlab system source code is both well structured and well commented but there is also a need for a programmer to read the code if it has to be up-dated! I would also stress that systems handbooks have to be up-dated which includes adapted to changes in hardware equipment (esp. when student wizards are the target group). (Pettersson¹)

The amount of money available for Ozlab maintenance clearly is limited and since it will most likely continue on as a university project this will still be an issue for the future Ozlab. Knowing this, the aspect of making Ozlab maintenance easy and time efficient should be a high priority when developing the future Ozlab.

Finally, one of the Ozlab assistants also points out how important flexibility and portability is, as well as a file format that can easily be imported and modified (Angulo⁴). The Ozlab assistant goes on to discuss portability and a possible cloud-solution for Ozlab.

Portability of Ozlab, as a cloud-solution, discussed with the Ozlab assistant interviewed would bring both pros and cons with it. It would enable globalization of Ozlab since testing would be made possible from anywhere in the world, but it would also prevent the wizard from controlling the test environment and how that affects the test person. It is not necessary for the wizard to be able to control the test environment in order to practice the Wizard-of-Oz technique, but we think the test results would become less reliable without knowledge of the test persons surroundings. If there is a possibility for the wizard to receive real time video and audio, as well as transmit real time audio during a test, this would probably make the test results almost as reliable as if the test were conducted in a lab. If this is not the case, there are other prototyping tools that would probably be more suitable for the particular test than Ozlab.

---

¹ John Sören Pettersson, originator, researching user and teaching user, email 1st of May, 2011.
5.4 Simplicity

Both the researching user and one of the Ozlab assistants advocates for a lot more simplicity when using Ozlab. The Ozlab assistant and the commercial user also state, that the learning curve of the Ozlab system is too steep.

“You want it as easy as possible, but as powerful as possible, which becomes some kind of impossible equation. The more powerful it gets, the more complicated it gets. And the easier it gets, the less opportunities one gets” (Molin3)

While the researching user wants to exchange opportunities for simplicity, the Ozlab assistant purely wants simplicity from the Ozlab system. “Simplicity before anything else is what I think, so what’s easy for the user is what should be prioritized.” (Lindström6). The researching user goes on to discuss that simplicity makes it easier for the naïve users, this is instead discussed above.

We believe that simplicity is needed to a certain extent. When using Ozlab some handling have to be simplified, and a lot of functionality and behaviours could be made more available to the users “Users should not have to wonder whether different words, situation, or actions mean the same thing. Follow platform conventions.”(Nielsen 2001). In short we believe that what need to be corrected are unnecessary difficult tasks from the current system that can easily be simplified. In particular when handling the system there should be fewer steps to take when going from an idea to a finished prototype.

As Nielsen states in his heuristic principle 3. User control and freedom, the users will often use system functions by mistake, because of this they need to be able to step backward and/or undo activities easily. Another thing that needs to be improved concerning simplicity is that the user in the current system is often forced to make detours in the system to solve some really trivial problems in the workflow.

5.5 Functionality

One of the Ozlab assistants and the commercial users thinks that a certain amount of functionality needs to be made available in Ozlab.

Some of the functionality that we often use in our applications, such as tree structures and lists often cause problems for us when we conduct user tests since it is very well defined how that interaction should work, they are also very hard to test. Simple things like radio buttons – which we always use in some way or another – tend to cause problems. I would like to have standard components pre-made because such components do not have to be in focus in our interaction tests. (Nilsson5).
One of the Ozlab assistants stated that there are a number of components that should be built into Ozlab, such as a simple way to make a drop list. Several users have stated that a certain amount of drag drop functionality to help create basic components would be useful.

More functionality, in particular finished components to add to a prototype would improve and simplify the making of Ozlab prototypes. One example of such functionality is supporting undo/redo and providing this in a simple and time-efficient way, this is mentioned in Nielsen’s Heuristic principle number 3. User control and freedom. Considering the standardized elements that the commercial user wants, we admit that there is a point there, but since we earlier in this paper say that there are better, simpler and more efficient tools to use for prototyping standard interfaces than Ozlab. Making sure that Ozlab can provide standardized elements is important, but it is not of high priority when deciding which solution is optimal for the redevelopment of Ozlab, since they can be developed with this in mind.

5.6 User solutions

The users have some ideas on what a solution should contain. The researching user is more concerned with future modifications, the commercial user is more concerned about reusing the prototypes for further development and one of the Ozlab assistants wants Ozlab to be more available when testing ideas and prototypes that are unconventional.

“So I would like to prototype anything in Ozlab, and just use the wizard of Oz technique. Be flexible in the tools. The room allows you to do the wizard of Oz technique.” (Angulo⁴)

The researching user states that Ozlab will have to be open for future modifications, so that it will never be out of date. This includes Ozlab being modifiable to such an extent that future systems and platforms that have yet to be invented can be tested (Molin³). The Commercial user wants an Ozlab that will enable and help the creators of prototypes to easily reuse the graphics and routines made for the prototypes. This will help the user to not have to do the work twice (Nilsson⁵). The Ozlab assistant clearly stated that Ozlab has not yet been used to its optimal capacity (Angulo⁴). What the Ozlab assistant wants to do with Ozlab is to test new technologies such as interactive table interfaces or even mobile phones that could be twist-and-bendable in the future. In short, he wants to test systems that can only be truly prototyped in a Wizard-of-Oz setting (ibid.).

We conclude that Ozlab will have to be open for future modifications. Not only will this improve the short-term solution, but will prepare Ozlab for a stable long-term solution that is extendable. Just by having this as a foundation for the development we can satisfy the researching user as well as the Ozlab assistant. When new technologies are invented, an Ozlab that can easily be extended will always be kept up to date. We also agree that future solutions such as table interfaces and bendable phones will be optimal to prototype for in a Wizard-of-Oz setting such as Ozlab. The idea from the commercial users concerning reusable material is
something we wholeheartedly agree with, there is no reason to use Ozlab if it means that one will have to double the workload. We also want to build upon this idea and say that an Ozlab prototype should be reusable in future projects. If components and images are created for an Ozlab prototype, then why shouldn’t it be easy to reuse these in other future Ozlab prototypes?

5.7 Collected student thoughts

The following list is composed from 3 different reports made by students concerning Ozlab, the points worth mentioning are shown below (Gustafsson et al. 2010; Kusuran et al. 2010; Oscarsson et al. 2010).

Primary requirements

- Right clicks made by the test person should be visible for the test leader. Because some functions in a prototype tested may require right click.
- Navigation made by the test person should be better synchronized with the test leaders’ view. Because the test leader should be able to follow more easily what the test person does. (Mainly a problem with the laptop version of Ozlab)
- Hotkeys for freeze/unfreeze, hide/show wait screen and lock/unlock moveable objects. Because as a test leader it is hard to follow the test person and hotkeys would make the job easier.
- A button menu to navigate in for the test leader. Because the only menu that the test leader can use is the drop down menu and a button menu would work better.

Secondary requirements

- Possibility to make animations on objects that runs when test persons click on the object. Because it would further minimize the pressure of the test leader.
- An own standing program for development of shells.
- Because Director is too complex to use for shell development and an own program could be much simpler and exist only of components that will be used in Ozlab.

Wishes

- Snap function when an object is dragged to an area.
- Because when test leaders need to move an object it may be hard to place it exactly where it should be and a snap function would place it at the exact position.
- Ozlab should be able to run on mobile devices, touch screens different platforms and Operative Systems.

It is quite obvious that the students’ primary concerns are from a test leader’s point of view and how to make it easier for the wizard while testing. This is most likely due to the fact that
this role is all they have experienced. Most of the requirements are about flaws in the user interface for the test leader. Everything that makes it easier for the wizard during tests is essential to keep Ozlab a viable option for the users. It is obvious that the student user group find the user interface of Ozlab unusable, this is most likely because of Director. To improve the User interface of Ozlab will thus greatly improve the user experience for the test leaders and make it more likely that Ozlab will meet the user groups’ demands and needs. The need for a better user interface is something that came up quite often in the interviews as well, it was especially brought forward by the Ozlab assistant (Lindström). Two of the student rapports gave some examples of how the future Ozlab user interface could look, as seen in Appendix B 1-2.

There could be an independent program to create the shells, a program entirely detached from Ozlab. Most likely the student users have noticed this by the problems they have faced when previously trying to create the shells in Director.
6. The Optimal Workflow

In the process of creating and maintaining an Ozlab prototype there are several activities that the future solution has to support. Fig 6.1 shows a model that describes these activities and how they are connected.

![Diagram](image)

Fig 6.1 Activities in the Ozlab prototype process.

The process starts with the creation of the graphical objects needed in the Ozlab prototype. These graphical objects can be anything from advanced computer made graphics, photos, hand-drawn sketches that are scanned in, video and text. When the graphics have been created, there has to be an opportunity to add Ozlab functionality to them. By this we mean the functionality that is required to run the prototype in Ozlab as described in section 4.2.2, under the heading “Template functions”.

When the images and functionality has been added to the Ozlab prototype it is ready to be used for testing. Even after the prototype is ready the system has to support editing of both functionality and graphics. The editing of functionality is limited to adding or subtracting the pre-programmed wizardry mentioned above. Both Functionality and graphics should be editable separately, but editing the graphics shall not undo previously added wizard functionality. This is an iterative process as shown in fig 6.1.
7. The Possible Solutions

In this chapter we present the solutions based upon the file formats presented in section 3.4 – Analysis of File formats. Solution models and workflows are presented for each solution to give a full description of it.

7.1 The Photoshop Solution

The reason why Adobe Photoshop is a feasible solution is that it allows easy making of images and allows these images to be reused. Although this solution would make Ozlab dependant on software, Adobe Photoshop will most likely be around for a long while since it is the most well-known image-editing software available up until 2011.

The solution is based upon the PSD file format which stores enormous amounts of data concerning all images created for the file. The images can easily be structured inside of Photoshop in a hierarchical order together with grouping of the images. In the layer structure which is used in Photoshop, the layers which contain images can be renamed to suitable names to make structuring easier. The solution discussed by Brundin (2011) means that the Ozlab behaviours of the images and layers would have to be decided and manually written into the name of the layers, so that Ozlab later could interpret these layer names and apply the corresponding behaviours.

The Adobe Photoshop solution would mean that Ozlab would not be independent of software, but will instead rely quite heavily on a specific program. Both graphics and functionality (by name) will be added directly in Photoshop (fig 7.1). This will then be exported as a PSD file, and this file will then be imported into Ozlab.
If this solution were to be implemented it would require large amount of programming since Ozlab would have to be able to utilize most functions of the Photoshop file format. This is unlikely to be an easy task to accomplish. The Photoshop solution would require more manual labour from prototype-makers since functionality would have to be written and assigned directly in Photoshop. This is most likely the biggest obstacle to actually implementing this solution.

![Photoshop workflow](image)

**[fig 7.2] Photoshop workflow**

The Photoshop workflow supports the optimal workflow described in chapter 6 to a major extent. It enables iterations when editing that the test leader wants. Editing graphics might force you to edit functionality in some cases but since the Ozlab prototype is only a PSD file, all editing is in the same file.

### 7.2 The HTML5 Solution

The most important reason why HTML5 is up to discussion as a future solution for Ozlab is that it would provide all the features that an Adobe Flash solution would, but without the software dependency. Why Adobe Flash has been discussed is further explained in section 3.4 – Analysis of File formats.
This solution is based upon software independence, the idea is to use an image editor of choice to create the graphical components needed for a specific prototype, and then use any software that can export to HTML5 to build your prototype using a HTML5-based template (Lamberg 2011). This template would contain all Ozlab functionality that is currently provided in the Director template-cast. The point of this is that as long as you use this template as a base for your prototype, you can use any software you like, from notepad(not recommended though) to an advanced HTML5 editor or a rapid prototyping tool to create your Ozlab prototype.

In order to make this solution work, the Ozlab Testrunner would have to be reprogrammed to behave similar to a web-browser, but with some modifications and some added Wizard-of-Oz functionality. This could cause problems similar to the ones that occur when some web pages work differently in different web-browsers. In that case the problem is that the browser interprets the code differently, so for the Ozlab Testrunner not to crash, it would have to be able to behave like any web browser existing today.

As mentioned before, an HTML5 solution like this one is software independent. However, it is depending on the HTML5 format instead, and the fact that the future version of it continues to be backward compatible, which has been a leading principle in the development up until now.
One of the things that could cause problems while using this solution is that the structure of the HTML5 code could differ a lot from one prototype to another, depending on which software is used to create them. For sure, all prototypes would be in the same format, but there could be differences between them anyway.

One way of minimizing this kind of problems could be to make the template very regulated, but on the other hand this would take away some of the software independence of the solution since the template would then always work better with a specific software then others.

To take this discussion one step further, a structured template could be a help in guiding the wizard to choose more appropriate software, rather than for example Notepad, because even if Ozlab Testrunner parses HTML5 files which makes it possible to use such a simple software, it is not preferable to actually use Notepad.

![fig 7.4] HTML5 workflow

Figure 7.4 shows the workflow of this solution, and we see that it differs from the workflow in section 6.1 considering the iterations when editing functionality and graphics. In chapter 6 we stated that the editing of graphics and functionality should be independent of each other, which is not the case with an HTML5 solution.

You can choose to edit the functionality in the prototype without changing the graphics, but if you edit your graphics, you will have to again add your functionality as well. Other than that, this solution supports all parts of the workflow in chapter 6.
7.3 The XML solution

Like the HTML5-solution, the XML-solution is based upon software independence. The idea is to use any image edit or to create the graphical components for the prototype and then import them into a shellbuilder created especially for Ozlab. In this shellbuilder the functionality is added and the output is an Ozlab prototype that consists of an XML-file and a folder containing the graphical components (fig 7.5). This means that instead of depending on specific software, this solution depends on the XML file format.

To make this solution possible it requires that an Ozlab shellbuilder is created that can import, create and export a XML file based on the images of specific file formats. This solution would denote a huge amount of programming and reprogramming since the Ozlab shellbuilder needs to be created from scratch, and the Testrunner accommodated to read XML. This also means that you give the opportunity to, in an easy way modify and develop Ozlab. If this solution is implemented it is therefore of great importance to be mindful of future modifications and maintenance. One thing that could be a problem with this solution is if XML drastically changes in future versions, but since both of the programs that handle the XML are tailor made for Ozlab, adapting Ozlab to future XML versions would not be a big obstacle.
Figure 7.6 shows the workflow for the XML solution. The iterations possible for editing the Ozlab prototype are the same as discussed in chapter 6. Because you have an XML file and a folder of images, it is easy to edit either and both of these. Since the XML file contains the paths to the images, if the images were edited without the XML being edited, it would still show the newer images. That is, the prototype can go directly back to testing without necessarily going through the stage “Functionality Adding & XML export”. So this solution will make the work for the test leader more flexible.
8 Recommendations

8.1 Comparison and evaluation

In this section we compare the solutions in chapter 7. We evaluate the solutions based upon how well they meet the requirements of the Ozlab User groups discussed in chapter 5, and how well they support the optimal workflow presented in chapter 6.

The XML solution is the only one that fully supports the optimal workflow. Both the Photoshop and the HTML5 solutions almost fully support the optimal workflow, they both provide all the activities, but not the independence between editing graphics and editing functionality. If you edit the graphics in the Photoshop solution you might have to edit functionality as well. In the HTML5 solution it is required that you edit the functionality if you have edited the graphics.

The HTML5 and the XML solution are both software independent. Instead they are dependent on a specific file format. The Photoshop solution is naturally depending on Adobe Photoshop, the PSD file format and its future development. Based upon our discussion in section 5.1, the software dependency of the Photoshop solution would be a bigger disadvantage for the Future Ozlab, than the file format dependency of the other two solutions.

Considering naïve users, we have stated that they should not be prioritized in the development of Ozlab, but we think that the solution most suitable for naïve users is the XML solution. The Photoshop solution requires a lot of prior knowledge from the user, and the HTML5 solution forces the user to choose an HTML5 editor, which most likely also requires prior knowledge. The XML solution only requires images and Ozlab software.

Long-term sustainability is one of the most important things to consider when developing Ozlab for the future, to prevent Ozlab from getting dated, which is the case today. The solution with most potential to be long term sustainable is the XML solution because it does not depend on any external software and therefore the Shellbuilder and the Testrunner will always be at the same level of development so one of them cannot make the other one dated. The problem with the other two solutions is that they are either depending on an external file format or software, which either could be developed or discontinued at any time and with no possibility for the people maintaining Ozlab to interfere.

Considering simplicity and in particular availability for users, the XML solution stands out. The Shellbuilder could quite easily be programmed to have all functionality and behaviour available to add to images with an easy drag and drop interface. The HTML5 solution could be made simple if a template was available with all necessary functionality. But on the other hand this removes a big aspect of the solutions software independence since the template would have to be tied up to specific software. The Photoshop solution is the one that is not all that simple for users to handle, since there would have to be a lot of manual programming to
get functionality to work. For users acquainted with Photoshop and programming this could be a simple procedure, but for most users it would involve a lot of unnecessary work.

When it comes to functionality the solution that has the conditions to be truly functional in both aspects, functionality and finished components, is the XML solution. Finished components can easily be added and made available in the Shellbuilder as well as functionality which also are available in the shellbuilder. This depends on how well the Shellbuilder is designed and programmed. The Photoshop solution makes it easy to have finished components available to use, but lacks availability in functionality. Like the XML solution as well as the HTML5 solution, it all depends on what functions the Testrunner is programmed to work with. The HTML5 solution enables both finished components and functionality in an easy-to-access way, but this also requires the use of an HTML5 template as previously discussed.

To make the usage of Ozlab more time-efficient, it is essential that the prototype and its content can be reused. If the user can easily access the images in the prototype and reuse these for the finished product, or use them as a part of the requirements specification it would help the users in the usage of Ozlab. All three solutions enable the reusing of the prototype. The Photoshop solution is so close to the image-creating process, that getting the images and editing them for other uses than prototyping is basically already involved in the process. However, the HTML5 solution as well as the XML solution is built upon images that are contained within the prototype, to extract these from the prototypes would be a simple process.

Since one of the users that were interviewed clearly expressed a wish that Ozlab should be used to test unconventional interfaces on new types of devices and not only GUI interfaces, it is relevant to see how the different solutions could enable testing of such interfaces. We believe that both the HTML5 and XML solution could provide a prototype that is flexible enough to be tested on all devices that the Testrunner is adapted to work with. Since the Photoshop solution’s prototype will be a PSD file, it will most likely not work on some devices. These will most likely work better with simple images.

Table 8.1 describes how the three solutions support the optimal workflow and how they meet user groups’ requirements.
This table compares the XML, HTML5 and Photoshop solutions against the criteria:

<table>
<thead>
<tr>
<th></th>
<th>XML Solution</th>
<th>HTML5 Solution</th>
<th>Photoshop Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimal Workflow</td>
<td>✓</td>
<td>X</td>
<td>O</td>
</tr>
<tr>
<td>Software Independence</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Naïve Users</td>
<td>✓</td>
<td>✓</td>
<td>O</td>
</tr>
<tr>
<td>Long-term Sustainability</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Simplicity</td>
<td>O</td>
<td>O</td>
<td>X</td>
</tr>
<tr>
<td>Functionality</td>
<td>✓</td>
<td>O</td>
<td>✓</td>
</tr>
<tr>
<td>Reusability</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Unconventional interfaces</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
</tbody>
</table>

✓ means that the solution fully supports the requirement
O means that the solution partially supports the requirement
X means that it does not support the requirement

8.2 Recommendations

We recommend that the definition of the XML solution described in 7.3 is used as a base for conducting future development of the Ozlab system. Table 8.1 clearly shows that this is the best solution out of the three presented in this paper. The XML solution accommodates all the primary user groups’ needs. Not only does it excel when it comes to long-term sustainability, software independence, functionality as well as meeting the preferred workflow of Ozlab users, it is also truly open for future modifications and improvements.
References


Appendix

A. Interview questions

What is your background/relationship with Ozlab?

How would you like to be able to use Ozlab in the future?

What would you like to use a prototyping tool for?

What would be best for you to work with in Ozlab? A file format that is dependent on a program, or a file format that is independent of programs?

Let’s say you go from graphical parts, to a complete Ozlab prototype. When during this process do you think it would be best to add functionality to the graphics, or parts?

How does your dream-scenario look when creating Ozlab prototypes? In which way, with what tools, would you like to be able to prototypes?

If you just think from your own perspective, what would you like to use?

What’s important to think about when you develop Ozlab, if you consider a long-term solution? So that Ozlab will be up to date for an extended period of time.

The Future users of Ozlab, will they differ from users of today?

Do you think that the demands of Ozlab should change significantly for new and future Ozlab users?

Do you think that prototyping in Ozlab will require more or less knowledge about User Testing and usability from future wizards?

Will the future usage of Ozlab differ from how it is used today?

Which file format(s) do you think would be a good alternative to build Ozlab prototypes in?
B.1 Ozlab GUI example 1

Design proposition (Kusuran et al. 2005)
B.2 Ozlab GUI example 2

Design proposition (Gustafsson et al. 2005)