Study of Prototyping Tools for User Interface Design

Xavier Righetti

January 27, 2006

Bachelor Thesis Supervisors: Prof. G. Falquet, Patrick Roth
Contents

1 INTRODUCTION 4

2 RAPID INTERFACE PROTOTYPING STATE-OF-THE-ART 5

2.1 What is rapid prototyping ? ............................. 6

2.2 Why use rapid prototyping ? ............................ 7

2.2.1 Key Issues ........................................ 7

2.3 Taxonomies ............................................. 8

2.3.1 Horizontal / Vertical : ............................. 8

2.3.2 Top Down / Bottom Up approach : ............... 9

2.3.3 Level approach .................................... 11

2.4 Prototyping Tools: ..................................... 14

2.4.1 Paper and Pencil .................................... 15

2.4.2 Facade Tools ........................................ 18

2.4.3 Interface Builders .................................. 20

2.4.4 Programming Languages ......................... 22

2.4.5 Summary ............................................. 22

2.5 Conclusion ............................................ 24

3 MANUAL FOR USING MACROMEDIA FLASH AS A UI
RAPID PROTOTYPING TOOL 24

3.1 Concepts ............................................ 25

3.1.1 Timeline .......................................... 25
1 INTRODUCTION

During the last 10 years, the development of highly interactive software with improved Graphical User Interfaces (GUI) has become increasingly common. User Interfaces (UI) have become more and more complex, providing a total freedom of creativity for the interface designer, but at a certain cost: the interface could be incomprehensible and the user could risk to lose himself during the interaction. For example, old ms-dos text-based applications were not so fancy, but at least they had the advantage of having a clear and understandable interface. Some of today’s applications or operating systems are so unnecessarily complex that their use can generate a lot of frustration.

We know that the factors of success of a software are directly related to its behavior facing the users’ actions and to the quality of the communication between them. It is therefore essential to have users (beta-testers) testing and commenting the interface from the beginning. This way, the developers are able to constantly re-aim the design of the UI even without having started the proper development phase. Thus there is a need to quickly produce and show mock-ups of UI to the user, so that we don’t loose time developing in the wrong direction. These mock-ups are called ”Prototypes”.

In this document, we will discuss about UI rapid prototyping. Chapter two will define the state-of-the-art and its critical importance for developing UI while chapter three holds a manual for using macromedia flash as a UI prototyping tool. Then, we will end with an experiment in chapter 4.
2 RAPID INTERFACE PROTOTYPING STATE-OF-THE-ART

Rapid Prototyping is a development method situated in the process of the User Interface Design (UID). During the stage of Rapid Prototyping, specialists use a different approach than the classical "waterfall cycle" \(^1\). Instead, this method uses the "spiral model" of development which is a continuous cycle described in the figure below:

![Spiral Development Method Diagram](image)

Figure 1. representation of the spiral development method

The Conception stage is where the concept definitions are being elaborated and / or refined. The Prototype stage is of course the development of a

\(^1\)The waterfall model is a software development model describing the naive approach to software development in which development is seen as flowing steadily through the phases of requirements analysis, design, implementation, testing (validation), integration, and maintenance.[1]
prototype, whereas the Evaluation stage contains the actions of showing the prototype to users and getting their feedback.

2.1 What is rapid prototyping?

Boar claimed that rapid prototyping is "a specific strategy for performing requirements definitions wherein user needs are extracted, presented, and successfully refined by building a working model of the ultimate system quickly"[3].

Rapid Prototyping always starts with a sketch of the interface itself. The goal is not to have a precise layout of the application, but instead to visualize the functionalities of the application and all possible tasks you can perform.

This methodology of creating mock-up interfaces for testing purposes offers us a way to precisely understand the needs of the user. Rapid Prototyping is therefore an essential iterative methodology of a User-Centered Design approach, where interfaces are created, evaluated, and refined until the desired usability criteria are achieved.

Prototypes can range from extremely simple sketches as you can see on figure 2 (low-fidelity prototypes) to full systems that contain nearly all functionalities of the final application (high-fidelity prototypes). But most of all, prototypes should be cheap, must be created in a short amount of time, and for large projects are often mandatory.
2.2 Why use rapid prototyping?

Rapid prototyping helps us to save time and resources. Indeed, a relatively cheap mock-up is built in one day to experience the same feelings you would have with a real system. The use of prototype authoring tools facilitates the identification of functional requirements during analysis and design phases, but the greatest limitation of this technique remains the identification of hardware or software problems and hazards. This technique may be used either to test quickly the consumer interest or the product’s technical feasibility. For example, the creation a paper prototype for a complex five-step subscription process will help us to minimize database accesses and usability problems.

Rapid Prototyping helps us to achieve two main goals:

1. reduce the cost risk,
2. ensure the quality of the final product.

Software developers are often confronted with users afraid of computers, uncomfortable with any applications and so unable to give suggestions. With the help of a tangible prototype, the trust of the user will increase, the relationship and the communication between the user and the developer will improve, resulting in clearer goals and tasks and thus more chances of acceptance of a project. On the other hand, a user might have many unrealistic expectations about the functionalities of the desired application, but the prototype would help him distinguish what is possible from what is not. In both cases, prototypes are our only way to learn from experimentation.

2.2.1 Key Issues

Before choosing the right prototyping methodology and authoring tool, there is a question that need to be asked: what kind of prototype do I want to build?

To help us answer to that question, Pedro Szekely [14] highlighted different characteristics of an application that need to be reviewed before actually running into the elaboration of a prototype.

First of all, we need to know who the user is, and the exact tasks he should
be able to perform with the software. This logical step helps focusing the development at the right direction. Then, we can concentrate on the abstract level of the interface, for example, the kind of content that will be displayed, the number of items to appear in a list, the type of inputs, and so on.

During the elaboration, we have to write down why several design choices were made so that we don’t forget the relationship between the technical constraints and the choice of objects type and placement. This principle is called the "Design Rationale".

Another key issue is the answer to the question: "do we want to reuse the code of the prototype or start-over for the real application?" It is sometimes easier and faster not to recycle the code.

### 2.3 Taxonomies

There has been various ways for classifying rapid prototyping development methods according to different authors. The three main taxonomies I retained are: Horizontal / Vertical [15], Top Down / Bottom Up [7] [18] [6], and Low / High Fidelity [7] [13].

#### 2.3.1 Horizontal / Vertical:

Horizontal versus Vertical designates two different approaches for developing a prototype. Each method has its own advantages and disadvantages during specific stages of the prototype development.

With a horizontal approach, the whole system is first built at a high level to check how it will cover the scope of the project. The details are filled in later, but all the main components of the interface are included. This gives the illusion of a fully functional prototype.

With a vertical approach, a specific part of the system is built incrementally until it is fully understood, then the next section is started. This way of selecting one or two primary tasks makes sure that a substantial part of the vertical functionality required for those tasks is supported. 'Substantial part’ includes screens, error messages, handling of unexpected input, defaults, robustness, ...
The figure shows the Horizontal approach as the design of all interface screens, while the Vertical approach deepens the functionalities and details of only one screen.

The horizontal approach is most useful for the early stages of the design. Its purpose is to test the overall interaction metaphor, so it includes common functions that the user is expected to perform frequently. On the opposite, the vertical approach is most useful in the later stages of design, to test details of the design.

2.3.2 Top Down / Bottom Up approach:

The use of the Top Down versus Bottom Up approach mainly depends on the number and the precision of the project’s requirements.

The Top Down approach analyzes the tasks to define a conceptual design and models the interface in increasing levels of detail. This frequent way of proceeding is appropriate for projects with functional requirements. Indeed, it can completely address the problem ensuring no items are missed.
On the figure, the prototyping development starts with the main screen A, then once validated, the development goes on with screen B, which is a subsection of A, and so on until the prototype has reached the desired depth level.

The **Bottom Up** approach is an iterative, trial-and-error method. Once a "best guess" of the interface has been prototyped and evaluated, a new "best guess" is redesigned and so on. This method is appropriate for projects without known functional requirements and when there is little understanding of the problem. It enables the problem to be addressed in small chunks. This approach is well suited for rapidly changing environments and more forgiving than the top down approach.

As a matter of fact, the Bottom Up approach is only used when it is impossible to clearly know the functional requirements in advance. Although rare, this approach may occur for small iterative processes like the suggestion of a new function. Wilson and Rosenberg[18] defined this distinction in 1988 but nowadays, most of the user-centered approaches use the Top Down approach.

To summarize, the top down approach is about "knowing it all" at the start, whereas the bottom up approach is about "learning it" along the way.

2.3.3 Level approach

There are two major approaches depending on the precision level of the prototype: low and high fidelity. The usual life of a prototype starts at a low-fidelity (paper prototype) to end at a high-fidelity (programming language).

A **Low-fidelity** prototype is a cheap, simplified mock-up of the UI used for evaluation and participatory design sessions. "Low-fidelity" means the accent has been put on the rapid delivery instead of producing a precise prototype. In this case that the prototypes you use don’t have to really look like the actual interface you’re testing, as long as they ”work” the same. Since low-fidelity prototypes are cheap, both in terms of money and time, you can afford to have more cycles of definition, implementation, testing and evaluation.

On the other hand, **High-fidelity** prototyping is a method where the prototype is implemented to be as close to the final product as possible in terms of look and feel, interaction, and timing. The prototype accepts input from the keyboard or mouse like the actual interface would, and responds to those events in the same way (displaying a particular window or message, changing state, etc.). High-fidelity prototypes should be used for teaching and marketing.

The figure below represents the increase of fidelity and effort, according to the progression of prototyping methods.
It is difficult to have a UI tool which is both easy to learn and functional. This is why we retained Daniel Engelberg’s hybrid solution called Mid-Fidelity Prototyping [7].

Mid-fidelity prototyping tools are used after early design, for the purposes of detailed design and usability validation. They present detailed information about navigation, functionality, content and layout, but in schematic (wireframe) or approximate form. Mid-fidelity prototypes serve as a reference for the functional specifications.

The table below summarizes the different appearance, optimal use, advantages and limitations of the three levels of fidelity.
<table>
<thead>
<tr>
<th>Fidelity</th>
<th>Appearance</th>
<th>Optimal uses</th>
<th>Advantages</th>
<th>Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Low</strong></td>
<td>Rough sketches, highly schematic and approximate. Little interactivity.</td>
<td>Early design: conceptualizing and envisioning the application.</td>
<td>Low cost: useful communication vehicle; proof of concept.</td>
<td>Limited usefulness after requirements established; limitations in usability testing.</td>
</tr>
<tr>
<td><strong>Mid</strong></td>
<td>Fairly detailed, provides simulated interactive functionality and full navigation.</td>
<td>Designing and evaluating most interactive aspects, including navigation functionality, content and layout.</td>
<td>Much lower cost and time as compared to high fidelity; detail is sufficient for usability testing.</td>
<td>Does not fully communicate the look and feel of the final product; serves as a living specification document.</td>
</tr>
<tr>
<td><strong>High</strong></td>
<td>Lifelike simulation of the final product. Highly functional, but the back end might be simulated.</td>
<td>Marketing tool; training tool; simulation of advanced or highly interactive techniques.</td>
<td>High degree of functionality; fully interactive; defines look and feel of final product; serves as a living specification.</td>
<td>Expensive to develop; time consuming to build (compared to other fidelity levels).</td>
</tr>
</tbody>
</table>

Table 1: differences between low-, mid- and high fidelity prototypes. [7]
2.4 Prototyping Tools:

Once we have chosen which method to apply for developing our prototype, we need to have the suitable prototyping tool by finding what our needs and what each tool can offer. As each prototype differs from another, there are hundreds of ways of conducting a good prototype to term. With experience, usability specialists know which method works best for a very precise kind of prototype. Let’s make a list of features a good prototyping tool should possess[14]:

- **Ease of Use.** You definitely want any tool to be easy to handle and quick to learn.

- **Fast Turn-Around.** The ability of making changes and instantly see the differences.

- **Extensive Control over Prototype Features.** A good prototyping tool should support a large variety of design possibilities. The interface controls, as buttons, switches and inputs should be fully customizable.

- **Data Collection Capabilities.** When interacting with the prototype, the prototyping tool should collect every action of the user. The collected data may include many forms: user feedback while interacting with the prototype, answers to questionnaires, video segments, complete interaction histories in logfiles.

- **Executable Prototypes.** To be faithful to real systems, the prototype should be working in the same environment.

- **Lifecycle Support.** It would be wise to keep your prototypes in case you need one or even a part of one. Parts of prototypes should always be reusable.

- **Team Design.** The prototyping tool should support collaborative group work.

- **Version Control.** The use of a version controlling system as CVS (Concurrent Versions System) is strongly advised to explore alternatives, rollback to previous design, and give precise feedback for a specific version.
We are going to review several rapid prototyping tool categories, from the simplest to the most complex. Each tool distinguishes itself by its advantages and disadvantages during specific periods of the prototype development. Firstly, paper prototyping which is, according to the level taxonomy, a method to produce low-fidelity prototypes. Then, the facade tools which generally allows mid-fidelity and horizontal development. They will be followed by the interface builders which are used to produce mid- and high-fidelity prototypes using either a vertical or a horizontal approach. At last, we will review programming languages which produce high fidelity prototypes including reusable code.

2.4.1 Paper and Pencil

Paper and pencil prototyping is complementary to other rapid prototyping techniques and almost mandatory in early stages of design. They produce low-fidelity prototypes which are used during early stages of design for conceptualizing and envisioning the application.

The method itself involves creating rough, hand sketched, drawings of an interface to use as prototypes of a design. More UI widgets\(^1\) such as checkboxes, text fields, and drop-down lists can be cut out of a sheet of paper and added to the interface to simulate the interactivity. These prototypes can then be tested on users by asking them to perform some tasks using the roughly sketched out interface, while designers observe their actions to gauge where they face problems, and what they like. Though seemingly unsophisticated, this method is very successful at discovering usability issues early in the design process.

Paper prototyping is the most widely used method for designing and testing UI. It offers great benefits in terms of saved time and cost since it enables developers to quickly draw interfaces and test them before they begin development. This also allows for easy and inexpensive modification to existing designs.

Not only Paper Prototyping is easy to use with an extensive control over design details, but it also is the perfect team prototyping tool because every-

\(^1\)A widget (or control) is a graphical interface component that a computer user interacts with, such as a button or a text box. Widgets are sometimes qualified as virtual to distinguish them from their physical counterparts, e.g., virtual buttons that can be clicked with a mouse cursor, vs. physical buttons that can be pressed with a finger.\[^1\]
A Disadvantage of paper prototyping is that it may lead to insufficient analysis: if it works on the paper, it doesn’t mean it works on the computer. Also, the users might expect the performance of the ultimate system to be the same as the prototype.

For more information, please refer to Carolyn Snyder’s book[2].

According to Brad Myers and James Landay[8], four distinct stages may often be required during paper prototyping. Their four-step how-to consists in:

1. **Concept Design:** This is where every possible approaches are sketched out. Then, the validity of each sketches is verified following the usability requirements and the goals agreed. At the end, the best approach is selected.

2. **Interaction Design:** In this step, the structure of the UI must be set by naming every screen, page or activity. Following the idea of
Affinity Diagramming [17], each screen’s name can be written on a Post-It note, and organized in clusters. The Post-It notes are then rearranged to simplify user tasks.

3. **Screen Design:** We use any computer graphic tool from Microsoft Paint to Adobe Photoshop for creating rough designs of the screens’ structure. These layouts are linked together and the first usability test is performed with a user.

4. **Screen Testing:** A user is asked to follow a realistic scenario on the sketched out screens. The developer explains him what happens, records every actions and comments of the user.

In 1995, James Landay and Brad Myers developed a tool called SILK (Sketching Interfaces Like Krazy). This electronic way of sketching preserves the important properties of pencil and paper: a rough drawing can be produced very quickly and the medium is very flexible. It’s goal is to counter paper prototyping’s main drawback: the lack of interactivity by combining classical prototyping tools like HyperCard, PowerPoint and Paper prototyping. With the use of one software, you are able to design, construct and test the UI.

![Figure 6. Storyboard of the use of Denim (formerly SILK)](image-url)
2.4.2 Facade Tools

Facade Tools allow designers to construct the UI’s screens without a real application behind. Of course, each tool has its own domain of applicability, but globally, this category of tools takes the advantages of Paper and Pencil, by adding the basic interactivity of an executable prototype: a response to inputs such as a click on a button. Because of their limited interactive features, Facade Tools are more adapted for creating UI prototypes that focus on the graphical interface design rather than on the interactive behavior.

Facade Tools create mid-fidelity prototype which are used to design and evaluate mostly the content and layout, but also some interactive aspects, including navigation functionality.

These tools don’t produce reusable code, but this weakness is also an advantage: it is the reason why they are so easy to handle.

We are going to review Apple’s Hypercard and Microsoft’s PowerPoint. They haven’t been primarily developed and used for rapid UI prototyping but prototype designers have changed these tools’ domain of application.

Hypercard

This application developed by Apple is getting old, but was one of the most important prototyping tool 10 years ago. It was reputed to be widely popular and extremely effective.

Hypercard’s main function was the construction of hypertext applications, with cards and stacks in a very easy way. The simple authoring mode allowed all members of the design team to be involved in screen design and simple behavior definition. They were separating UI screens into a stack of cards and the user would be able to navigate through them. It also included HyperTalk, a powerful and easy to use programming language to manipulate data and the UI.

Hypercard had three modes of execution: end-user, simple authoring, application developer. It was adapted to build WIMP\(^1\) Interfaces. It’s main drawbacks are its execution speed and its narrow domain of applicability.

\(^1\)WIMP: Windows, Icons, Menus, Pointer
**PowerPoint**

As everybody know, PowerPoint is a Microsoft application for creating presentations. Extremely easy to handle, this tool has also limited functions. Prototype developers use this tool just like Hypercard: they place one UI screen per slide and let the user navigate through them. If we look at its advantages compared to other prototyping tools, PowerPoint enables you to present the prototype everywhere (most people will have powerpoint in their office). You can create templates for every slides with the master slide, arrange slides between other slides, and even save as a series of web pages with one click of a button.

A prototype developed with PowerPoint remains just like a slide show presentation: each slide shows a fixed screenshot, which you can modify in a computer graphic program like Photoshop and import, or which you can modify directly in PowerPoint. Furthermore, the storyboard doesn’t have to be linear: you can create hyperlinks that jump to any slide in the presentation.

Powerpoint’s main advantage is it’s simplicity of use: everyone can prototype! That’s why it remains the most used prototyping tool.

For more information, you can find a small tutorial for using Microsoft PowerPoint 2004 as a rapid UI prototyping tool in chapter 6: Appendices.

**Macromedia Flash, Director**

Macromedia Flash and Director are powerful vector based development tools. Compared to Flash, Director is a more complex multimedia authoring tool. Adapted to large graphical components, it is an effective tool for producing impressive simulations, visualizations, presentations and of course for prototyping UI.

Director is slightly more difficult to learn than flash, mostly because it has more features. But since the latest version 8, Flash almost caught up with Director. For example, Flash’s programming language called Actionscript 2.0 has become as powerful as Lingo, Director’s equivalent.

Flash and Director can produce any kind of prototypes, from low to high fidelity, including a full Flash-based application, depending on the depth of
the work on the prototype.

Schematically, Flash has the same structure as Hypercard: it uses symbols (stacks) which are filled with frames (cards). You can assign actions to symbols or frames using Actionscript (hyperscript). Parameterized symbols can also be reused as separate objects. They are called components.

In chapter 3, we will discuss how to use the basic functions of Flash to develop rapid prototypes.

2.4.3 Interface Builders

GUI builders are software engineering tools developed to increase the productivity of UI development teams, and to lower the cost of UI code in both the development and maintenance phases. One study found that an average of 48% of application code is devoted to the UI, and 50% of the development time required for the entire application is devoted to the UI portion [11]. They are the construction tools that give interface developers a drawing-like interface to specify the layout. Their main function is to save time by generating the executable code of the interface from a few mouse clicks.
A the present, almost each application development environment has its own interface builder. I chose the NeXT Interface Builder as an example because it is a famous prototyping tool that allows programmers to develop non-trivial application UI in a very short amount of time.

**NeXT Interface Builder**

The NeXT Interface Builder was created in 1988[16]. It provides a collection of UI objects (buttons, textfields, sliders, etc.) that a designer can select and drag into the application project window. These objects can then be connected to application software, but more expertise and effort is required to develop object-oriented applications required for interface to the Interface Builder.

There are three major shortcomings: They can construct only static Portions of an interface (such as menus, etc.). It is also difficult to isolate the
interface from the rest of the application (call-backs): no ease of use, no fast turn around, poor simulation of a portion of an application that have not yet been implemented. Finally, they force the developers to select concrete building blocks to specify interfaces even though the designer is not ready to choose a specific technique.

2.4.4 Programming Languages

Most programming languages will require a good understanding of variables and functions. A high level of programming knowledge is required to be sure to produce an evolutionary prototype in few hours.

Visual C++ was developed for Microsoft’s Windows operating system, which positioned it for a different market than the HyperCard users on the Macintosh. This part of Visual Studio software development suite is built around an improvement of the C language called C++, in conjunction with a direct-manipulation interface builder. The designer is able to create and place standard interface elements, set and modify their properties, and associate them with codes that execute when they are invoked by a user of the interface.

A typical Visual C++ application does not have the originality of many HyperCard stacks or Director prototypes, but instead operates within the normal interface conventions for Windows applications. The programming environment makes it easy for the designer to convert a Visual C++ project into an executable file that can be distributed and run by users who do not have the Visual Studio Environment on any computer that runs Microsoft Windows.

Visual C++ is the good prototyping tool if you already know the language and want to build an executable prototype that runs under Windows and has reusable code. But for programming beginners, it is much simpler to use a Facade Tool that lets you cycle through different interface screens.

2.4.5 Summary

The use of these tools greatly reduce the duration of the prototyping phase which itself reduces development time. Except for paper prototyping, an additional cost of use is the level of human expertise required to master the
supporting development tools, along with the time necessary to implement a software prototype.

As you can see on the table below, each of these cited tools can be used depending on the prototype’s requirements.

<table>
<thead>
<tr>
<th></th>
<th>Paper and Pencil</th>
<th>Programming Languages</th>
<th>Facade Tools</th>
<th>Interface Builders</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ease of Use</strong></td>
<td>*****</td>
<td>*</td>
<td>****</td>
<td>****</td>
</tr>
<tr>
<td><strong>Fast Turnaround</strong></td>
<td>*****</td>
<td>*</td>
<td>*****</td>
<td>****</td>
</tr>
<tr>
<td><strong>Extensive Control</strong></td>
<td>*****</td>
<td>*****</td>
<td>****</td>
<td>****</td>
</tr>
<tr>
<td><strong>Data Collection</strong></td>
<td>*****</td>
<td>*****</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td><strong>Executable Prototypes</strong></td>
<td></td>
<td>*****</td>
<td>***</td>
<td>****</td>
</tr>
<tr>
<td><strong>Lifecycle Support</strong></td>
<td>*****</td>
<td>*</td>
<td>****</td>
<td>**</td>
</tr>
<tr>
<td><strong>Team Design</strong></td>
<td>****</td>
<td>**</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td><strong>Version Control</strong></td>
<td>*</td>
<td>***</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td><strong>example tool</strong></td>
<td>SILK</td>
<td>Visual C++</td>
<td>Flash</td>
<td>NeXT</td>
</tr>
</tbody>
</table>

Table 2: Summary of the Prototyping tools categories and their requirements [14]

In an old survey from July 2002 (cf. Table 2) conducted by The Interaction Design Group (guuui.com), people have been asked: "What is the primarily type of prototyping tool you use for prototyping?". None of them answered Macromedia Flash or Director.

There may be few potential reasons for this bad score. First, the survey was conducted in 2002; flash was less popular than nowadays. The question was about web prototyping tools usage; it seems people are reluctant to prototyping a website in flash, and changing it to html afterwards. Flash seems to be too difficult to use. A good solution might be to use a tool like Snapp MX to produce flash content in a much more easy way.

More information about Snapp MX: snapmx.com
Table 3: "What is the primarily type of prototyping tool you use for prototyping?" (52 replies)

<table>
<thead>
<tr>
<th>Tool Description</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-computerized tool such as pen and paper (paper prototyping), white boards, Post-It notes and overheads</td>
<td>7.5 %</td>
</tr>
<tr>
<td>Presentation software such as PowerPoint from Microsoft</td>
<td>9.4 %</td>
</tr>
<tr>
<td>Graphic design tool such as Illustrator or Photoshop from Adobe</td>
<td>18.9 %</td>
</tr>
<tr>
<td>Diagramming tool such as Visio or SmartDraw</td>
<td>24.5 %</td>
</tr>
<tr>
<td>Visual or text based HTML tool such as FrontPage, Dreamweaver or HomeSite</td>
<td>28.3 %</td>
</tr>
<tr>
<td>A software development tool such as Microsoft Visual Basic or Borland Delphi</td>
<td>1.9 %</td>
</tr>
<tr>
<td>A multimedia tool such as Director from Macromedia or Authorware from Macromedia</td>
<td>0.0 %</td>
</tr>
<tr>
<td>Other</td>
<td>9.4 %</td>
</tr>
</tbody>
</table>

2.5 Conclusion

This chapter introduced the rapid prototyping method. We defined the state-of-the-art by reviewing the different development taxonomies and tools. To our point of view, the most important classification is the level approach, which specifies the depth of precision of a prototype.

We reviewed different tool categories according to various taxonomies. Macromedia Flash has been retained as a powerful facade tool capable of rapidly developing prototypes. Next chapter will deepen the use of flash as a rapid prototyping tool.

3 MANUAL FOR USING MACROMEDIA FLASH AS A UI RAPID PROTOTYPING TOOL

This chapter will explain several concepts of Flash which will be required to use this authoring tool. Then, we will give you a methodology to conduct a good interactive prototype to term.

This manual will not cover the topic of shape morphing and traditional
animation techniques described in many books. Instead, a special attention will be paid to the interactivity and the non-temporal playback.

3.1 Concepts

This section will present the most important concepts to be aware of when starting to work with Flash. Each of these concepts are tied together. For example, the timeline contains the frames which contain symbols or components.

Main Scene → Timeline → Layers → Frames

Components ← Symbols

Movie Clip Button Graphics

Figure 8. This schema represents the relation of the different Flash elements, from the main structure to the content elements.

3.1.1 Timeline

The timeline organizes and controls the document’s content over time in layers and frames. Like films, Flash documents divide lengths of time into frames. Layers are like multiple film strips stacked on top of the others, each containing a different image that appears on the Stage.
Figure 9. On this figure, the timeline holds two layers. Each of the layers contains two frames. The vertical red line represents the frame on which we stand.

### 3.1.2 Frames and keyframes

A **keyframe** is a frame in which you define a change in an animation or include frame actions. Flash can tween\(^1\) the frames between keyframes to produce fluid animations.

The order in which frames and keyframes appear in the Timeline determines the order in which they are usually displayed in the application. You can arrange keyframes in the Timeline to edit the sequence of events in an animation.

A frame holds the same content as the previous frame, while a keyframe holds new content. A blank keyframe holds no content.

### 3.1.3 Symbols

**Symbols** are the greatest strength of Flash. They allow the reuse of graphics, buttons or movie clips without significantly increasing the file size. Once an object is converted to a symbol, it is automatically placed in the movie’s **library**. Symbols can be dragged from the library and placed on the stage. This creates an **Instance** of the symbol. Instances can be considered as

\(^1\) Short for “in-between”, and refers to “shape tweening” and “motion tweening” processes, where the user can define two key frames and Flash will automatically create the in-between frames, either morphing one shape into another over a set period of time or else moving a shape or shapes from point A to point B over a set period of time.
a copy of the symbol. The basic method to add a new symbol is to add something on the scene, then select and convert it to a symbol. You can now reuse as many instances of this symbol as you like.

![The library holds the different symbols contained in the flash movie.](image1)

**Figure 11.** The library holds the different symbols contained in the flash movie.

### 3.1.4 Components

**Components** are movie clips with parameters that let you modify their appearance and behavior. A component can provide a wide range of functionalities. A component can be a simple user interface control, such as a radio button or a check box, or it can be a complex control element, such as a media controller or a scroll pane.

Components allow you to quickly build applications with a consistent appearance and behavior. Their main drawback is that they require a bit of Actionscript 2.0 to make them work.
3.2 Guidelines for developing rapid interactive prototypes

We divided the UI prototype development in three stages: the conceptualization where the structure of the flash movie is being elaborated, the creation of the static content, and finally, the addition of interactivity.

3.2.1 Conceptualization

Before jumping into the prototype development, we first need to analyze the tasks we would like to perform. This will help us to develop a list of requirements for the application. The best solution to do so is to arrange a workshop attended by the main users and the developers of the future program.

Next step is really important although often omitted: the creation of a quick paper prototype (cf. section 2.4.1). This helps for revealing the structure of the application and prioritizing what should be present in the UI prototype.

3.2.2 Static content

Once the paper prototype has been validated, we use Flash to build a static, non-functional prototype that just includes the key UI elements, like textfields, buttons, etc. At this stage, we use flash just like PowerPoint that is, to have one interface screen per frame and the ability to navigate through them.

The creation of symbols is an important step as they represent the main structure of the flash movie. For example, you want to build a prototype with a navigation menu on the left and some content on the right (cf. figure 12). In this case, you need at least two symbols: One for the menu and one for the content. Therefore, the menu and the content remain independent.
You can also drag UI components from the components window to improve the look and feel of your prototype. Components are like prebuilt movie clips that make your work easier and faster. They are present to facilitate the creation of UI, but require some knowledge of Actionscript 2.0 to be able to play with them. It would be way too fastidious to create your own components only for a prototype.
3.2.3 Interactivity

Once our flash document’s structure is approximately completed, we need to add some interactivity. This can be done by selecting buttons on the scene and assigning actions to them through the Action window.

![The action window](image)

Figure 14. The action window

The code written below tells the flash player to go to frame number 3 when the mouse clicks on a button.

```actionscript
on (release) {
    gotoAndStop (3);
}
```

For example, we need a Yes/No popup. The idea is to create a two-state popup object that can be reusable at any time in the UI. We will use a movie clip symbol with two frames. The first frame is empty but the popup content stands on the second frame. The last step is to add few line of actionscript to show or hide the popup by going to the first or the second frame of the popup movie clip.
Once the user has clicked on yes or no, the popup has to disappear. The following code is the corresponding action of the yes — no button:

```javascript
on(release){
gotoAndStop(1);
}
```

Follow an iterative process of building a simple version of the application, getting client feedback, adding functionality, getting more feedback, adding more functionality and so on, until the prototype has reached the desired level of fidelity or until the flash application is complete.

Always keep in mind the goal you wish to attain. Choose the solution that leads you to this goal within the minimum of time. Often, the simplest solutions are the best. For more information, please refer to chapter 6: Appendices for a detailed Flash’s How-to.
4 EXPERIMENT

In order to verify Flash’s effectiveness for creating a rapid prototype, we organized a little experiment. A very simple prototype of an address book was created. We asked different people to reproduce the prototype using either Flash or PowerPoint. At the end, we analyzed the different results to compare both authoring tools as rapid prototyping tools.

4.1 Participants

The participants in our experiment consisted in 10 persons. Their skills ranged from vague idea to a good knowledge of the tool they were assigned to. I assigned one prototyping tool to each group (G1 and G2), with 5 participants per group. G1 used Macromedia Flash while G2 had to use Microsoft PowerPoint. Each group were asked to design the same prototype using the prototyping tool they were testing. Upon completing all tasks, each participant was asked to give his or her personal comments about the technique they used.

4.2 Methodology

All participants were given half an hour to familiarize themselves with the prototyping tool. The next day, during the testing period, no hints were given. The participants had to build the same prototype with their assigned prototyping tool. Task completion times and the participants’ comments were noted down. A grade ranging from 1 to 6 has been attributed to each prototype depending on their similarity to the original prototype.

4.3 Tasks

We chose to base the development to a medium level of interactivity: buttons trigger actions, pop-ups appear and disappear, the interface’s screens change. Since each participants couldn’t spend many hours for this experiment, the interface has been very limited.
Each participant was given the final prototype on paper with some explanation about the requested interactivity. They were asked to reproduce the exact layout and functionality. Finally, they had to fill out a post-experimental questionnaire with suggestions about the tool they used.

![Figure 16. The three UI panels of the reference prototype](image)

**4.4 Hypothesis**

We assumed that it is faster to build a prototype with Flash than with Powerpoint, mostly because of its ease of use and its adapted functionalities. But before starting the actual development of the prototype, the designer should know how to use the basic functions of Flash. We estimate the learning of this basic knowledge to about 1 hour.

Since most of the people are more familiar with Microsoft PowerPoint than Macromedia Flash, it seems obvious at first sight to think that the same task will be achieved more easily with PowerPoint. But none of these tools have been designed to develop prototypes. Hence, the results should differ greatly depending on the difficulty level of the requested prototype.
4.5 Results

As you can see on the figure below, the prototype created with PowerPoint has unexpectedly changed the color and style of some hyperlinks. This could be very annoying when trying to create a specific high-fidelity prototype whereas, most of the participants who used Flash did add interactive checkboxes and dynamic textfields (cf. figure 18).
Figure 18. Example of a prototype created by a participant with Flash

![Prototype Example](image)

Figure 19. Resemblance to the original prototype graded from 1 to 6 compared to the duration time of creation

![Graph](image)

The results showed that the use of PowerPoint is approximately 20% faster than Flash for prototyping the same interface. This slowness is a very serious
drawback for a rapid prototyping tool.

Flash seems to produce more detailed UI than PowerPoint. For example, none of the PowerPoint prototypes had working checkboxes or input textfields: this tool doesn’t support these kind of features which may be quite important depending on the requested fidelity level of the prototype.

![Notes Attributed by the Participants](image.png)

Figure 20. Grades attributed by the participants the during the evaluation questionnaire

According to the answers written on the post-experimental questionnaire and our observations during the experiment, the participants preferred to work with Flash. This seems contradictory since the results show that they had more trouble using Flash than PowerPoint.

The participants had almost always the same problems on both tools. Indeed, almost all participants had trouble with the selection of objects. For example, making the difference between the selection of an object and the selection of text or selecting an object behind another one.
4.6 Discussion

It is difficult to draw serious conclusions about this experiment since the participants had different skills, little time and were not many. Nevertheless, we can clearly see that Flash is slower for creating this little mid-fidelity prototype. But, maybe if the participants had been given another small prototype to develop reusing parts of their first prototype and the knowledge they acquired, the prototyping durations for both tools would have been close. Obviously, the advantage of reusing UI objects hasn’t been perceived by the participants because of the small size of the experiment’s prototype.

The main drawback of Flash is that rapid prototyping is not its main function, thus it contains too many superfluous functions that complexes the interface. Flash may be the best interface prototyping tool as long as the designer already knows how to use the software. This can be achieved in one week of intensive learning.

It should be used to produce rich interactive applications by graphic designers with no programming expertise. But, according to this experiment, the tool is not so ‘natural’ to learn in spite of the metaphorical film aspect Macromedia brought into Flash.

4.7 Conclusion

We imagine that the results would have been different if the participants had been given half a day of practice and more time to develop a larger prototype. Indeed, Flash is adapted to larger projects than PowerPoint. Because of its advanced features and its precision in UI design, Flash seems to be a great authoring tool for producing high-fidelity UI prototypes with advanced features such as error handling or data manipulation. But for the development of low- and mid-fidelity prototypes, I would recommend an easier tool to handle like PowerPoint.

To summarize, the use of Flash is recommended when:

1. you already know how to use it,
2. a high-fidelity prototype is required,
3. a non-trivial prototype with numerous functions to simulate is required
4. the amount of work to spend on the prototype exceeds approximately 20 hours,

5. you need to reuse several UI objects, for example a Yes/No dialog.

5 CONCLUSION

We saw that the factors of success of a software are directly related to the usability of its interface. A common solution to this problem known in many fields of interest is called rapid prototyping. This document tried to define rapid prototyping by explaining its function during software development. Rapid prototyping is a cyclical process that is held in three stages: conception, prototyping, evaluation.

We reviewed different taxonomies and retained the most obvious and essential classification: the level approach.

We also listed the different features a decent prototyping tool should possess and classified the tools into different categories: Paper and Pencil, Facade Tools, Interface Builders and Programming Languages. For each category, when gave an example tool and the kind of prototype it was adapted to build.

Then, we focused on the use of Flash to rapidly create prototypes. The main concepts of Flash were explained and connected together, so the reader could understand our Guidelines for developing rapid interactive prototypes. We also wrote two precise tutorials (cf. Appendices) about the use of Flash and the use of PowerPoint to rapidly create interactive prototypes.

A little experiment was organized in order to compare Flash and PowerPoint as prototyping tools. Five people used PowerPoint and five other people used Flash to produce the same prototype. The results did not fully satisfied our basic assumptions. Indeed, we imagined that Flash was faster and thus more adapted than PowerPoint, but the experiment proved the opposite, at least for prototyping very small low- or mid-fidelity prototypes.

It would be interesting to organize a larger experiment with at least 20 participants. We would train them for one week and then, the participants would have to build a high-fidelity prototype of a more complex UI under a day. This second experiment would test the capabilities of the authoring
tool under more realistic circumstances.

6 ACKNOWLEDGEMENTS

Many thanks to Patrick Roth for his remarks and proof reading, to the experiment’s participants for their precious time, to my family and my girlfriend for their support.

References


7 **APPENDICES**

7.1 **Macromedia Flash’s How-to**

1. Hit **Cmd+N** to start a new document,

2. use the **Text Tool (T)** to create the title, then change its style in the ‘Properties window’ (**Cmd+F3**) to make it look like the figure on page 24,

3. do the same for every text elements, don’t worry about the checkboxes, we will add them later,

4. use the **Line Tool (N)** to structure the content,

5. use the **Black Arrow (V)** to select objects on the scene,

6. hit Shift when you want to paste an object at the exact same place (**Cmd+Shift+V**),
7. hit Shift + an arrow when you want to quickly move the object with the keyboard,

8. please remember that you always have the possibility to go back to any previous step by hitting `Cmd+Z+Z+....+Z`

9. hit `Cmd+Enter` (test movie) and you should have something like this first panel:

![Image of the first interface screen](image-url)

Figure 21. The first interface screen (panel): the data-view

In the next steps, we will add some interactivity by creating invisible buttons.

1. Hit `R` to select the Rectangle Tool,

2. click on the `'Stroke Color' (1.) and select the `'no color' icon (2.) as indicated in the screenshot below,
3. draw a rectangle over the "Add new person" text,

4. select the rectangle by clicking on it,

5. convert it to a button by hitting **F8**. In the 'convert to symbol' popup, choose '**button**' and name it 'Invisible btn'

   You can see the rectangle’s border has changed to blue; it means it is now a symbol.

1. Go inside the button symbol by double-clicking on it,

2. click once on the first frame of the button to select it,

3. drag the first keyframe to the fourth frame.

   Now the button has no visual appearance, but only a clickable zone.

4. Go back to the main scene by clicking on Scene 1 in the top left corner as indicated in the screenshot below,

5. clone the button using the shift technique I described before (Cmd+C then Cmd+Shift+V to clone),

6. use the **Transform Tool (Q)** to adjust the size of the buttons,

7. to improve the look of the interface, you can drag checkboxes from the UIComponents window. Clear the label of the checkboxes from the parameters tab in the properties window.

   If we need to design several interfaces screens, the easiest way is to create one screen per frame. For example, the data-view screen on the first frame and the data-input screen on the second frame.
1. Hit **F6** to create a second frame on the timeline,

2. modify the content of the second frame to make it look like the second panel (input-screen),

![Figure 22. The second interface screen (panel): the data-input](image)

3. to add input textfields, draw a normal text box of the desired size (**T**), then inside the parameter window (**Cmd+F3**) choose 'Input text' instead of 'Static text' and click on the 'show border around text' button as indicated in the screenshot below,

![Figure 22. The second interface screen (panel): the data-input](image)

We are going to create a third layer which will contain our alert message.

1. left-click on the first frame to select it, then right-click on it and select 'copy frames',

Information System Interfaces University of Geneva
2. left-click on the third frame to select it, then right-click on it and select 'paste frames',

3. select the rectangle tool and draw the popup in the middle of the scene,

4. drag the Invisible_btn from the library (Cmd+L) on the popup and transform its size (Q) if needed. It should look like this:

![Image of the popup](image.jpg)

Figure 23. The third interface screen (panel): the data-view with a Yes/No dialog box

We now have all the necessary elements of our prototype. The last step is the coding part to setup the interactions.

1. click on the first frame, hit F9 to get the action window and type:

   ```javascript
   stop();
   ```

2. Simple-click on the first green button of the first frame (Add new person) to select it and type:

   ```javascript
   on(release){
   this.gotoAndStop(2);
   }
   ```
3. Copy this code, then simple-click on the second green button of the first frame (Cancel) to select it. Paste the code and change the number from 2 to 1:

```actionscript
on(release)
    this.gotoAndStop(1);
}
```

4. Click on the third frame to select it, then click on the green button, hit F9 to get the action window and paste the same actionscript code:

```actionscript
on(release)
    this.gotoAndStop(1);
}
```

5. That’s it!
Hit **Cmd + Enter** to test the movie.

### 7.2 Microsoft PowerPoint’s How-to

The first slide will contain our data-view panel, the second slide will be identical as the first one, but with a popup in the middle of it. Finally, the third frame will hold the data-input panel.

1. Select everything (**Cmd + A**) on the first slide of your new presentation and delete it,

2. To add some text, go to the **Insert** menu and click on **Text Box**,

3. To make it look like the original prototype, modify the text options using the **Font Pane** located in the **Formatting Palette** (View / Formatting Palette),

4. Proceed the same way for all text objects contained in the first slide,

5. To draw the structure lines, go to the **Lines** tab in the **Add Object** pane of the Formatting Palette,

6. Duplicate the first slide by selecting it in the left colon and by hitting **Cmd + D**,
7. in the second slide, add a rectangle box by selecting the ‘AutoShapes’ tab in the ‘Add Object’ pane of the Formatting Palette,

8. add text to make look like the popup window on the figure xx,

9. duplicate the second frame by selecting it in the left colon and by hitting \texttt{Cmd + D},

10. in the third frame delete everything but the title and the structure lines,

11. add the eventually missing elements to make it look like the input-view panel.

Now that we have all graphical elements, we need to add the interaction.

1. Go to the first frame, select the ‘Add new person’ object (be careful not to select the text but the object itself),

2. in the ‘Hyperlink’ pane of the Formatting Palette, Choose ‘Slide...’ and number three,

3. select the ‘Delete selected person(s)’ object and link it to slide number two,

4. go to the second slide, select the yes/no text box from the popup object and link it to the first slide,

5. go to the third slide, select the ‘Cancel’ object and link it to the first slide.

The prototype is almost ready. For the last step, we are going to disable the mouse clicks everywhere else than on our links.

6. Go to the ‘Slide Show’ menu, click on ‘Slide transition...’, deselect ‘On mouse click’ and click on ‘Apply to all’.