Technical Requirements/Criteria for UI Prototyping and Implementation Environment

SEED: Design for Usability Project
Project Members: Sheng-Fen Chien, Shang-chia Chiou, Bongjin Choi, Rana Sen,

Engineering Design Research Center
Carnegie Mellon University
Pittsburgh, PA 15213
schien+@cmu.edu
(412) 268-6271

1 Introduction

This document describes the technical requirement / criteria for the UI (User Interface) implementation and prototyping environment. It is intended that this document be used by all future developers of SEED related graphical user interface. Chapter 2 describes the technical requirements and criteria established for the selection of UI implementation and prototyping environment. Chapter 3 will presented the tools chosen for evaluation based on availability and recommendations of the DFU group.

2 Technical Requirements/Criteria

We decided not to consider high-level conceptual criteria, but rather decided to consider the technical requirements and criteria specific to the established or expected needs of the SEED environment. Separate requirements / criteria have been established for the UI prototyping environment and the UI implementation environment. The reason behind this decision is simple. The envisaged functionality of SEED is novel. Consequently, much iteration of the user interface and interaction design are anticipated. We would like to separate the short-term UI prototype from imposing on the long-term actual UI implementation. The following two sections outline the requirements / criteria selected for the respective environments.

2.1 UI Prototyping Environment

UI prototyping environments refer to UI builder and any environment that may support the rapid prototyping of the user interface including a mock-up of the interactions. The requirements / criteria for the prototyping environment are listed in their of order of importance. Each requirement / criterion are accompanied by a brief description

1. **Easy to learn and use**. The environment should require minimum time for the users to familiarize themselves. Its interface should be easy to use.
1. **Support incremental modification**. The environment should support as much as possible the expected iterations in design. For instance, interface layouts and color and widget spacing may be changed often. A good example of incremental modification support is the packer in Tk. The packer is a smart geometry manager of widgets. The user need not explicitly specify the geometry and placement of the widgets, but rather only specify their relative positions.
2. **Support interaction simulation**. The environment should have facilities to support interaction simulations. This may be provided via a high level language such as Hypertalk of Hypercard or some other fancy techniques.
3. **Support direct widget manipulation**. The environment should provide direct widget manipulation such as drag-n-drop.

4. **Easily generate complex/compound widgets**. The environment should provide support for easily generating complex/compound widgets. Such widgets will facilitate reuse of interface components.
5. **Free**. Last, the environment should be free. Since UI prototyping would most likely be a short-term endeavor, any capital investments are not justified.

2.2 UI Implementation Environment

UI implementation environments refer to the interface implementation language and widgets sets. It may also refer to any UIMS or UI builders that support the generation of application linkable code. The requirements/criteria for the implementation environment are listed in their of order importance. Each requirement/criterion are accompanied by a brief description.

1. **Easily generate/maintain complex/compound widgets**. (i.e., reuse)
2. **Support objects and inheritance**. (data structure)
3. **Support Model-View-Controller paradigm**. (i.e, event notification and separation of interface from the domain)
4. **Supports advanced interactions**. (e.g., drag-n-drop)
5. **Portable across Unix platforms**.
6. **Inexpensive**.

- **Interpreted versus compiled environment**. (turnaround time and execution speed)
- **Support incremental modification**.

3 Environments Considered and Evaluation

3.1 UI Environments

Environment	Type of Supports	Language
Alpha	UIMS /wo UI Builder	C++
ET++	Framework /w class library	C++
Hypercard	Prototyping tool	Hypertalk
iLog	Framework /w class library	C++
Interviews	UI Builder /w 2-D Xlib widgets	C++
Tk/Tcl	Interpreted toolkit	Tcl
Xf	UI Builder	Tcl
Winterp[a]	UI Builder	xlisp
Wxwindows	UI Builder	C++

Environment	Type of Supports	Language
Zinc	UI Builder	C++
Escarante[b]	Visual programming environment	C++

a. xlisp

b. modified version of ET++

3.2 Evaluation

	alpha	iLOG	Inter Views	XF	Wx-windows	Zinc	Escalante	Tk/Tcl	WIN-TERP	ET++	Hyper Card
learn / use	✔		▲	✘	✘	▲		✔	✔		✔
incr. mod.	✔		▲	✘	✔	✔		✔	✔		✘
interac-tion	✔			✘		✔		✔	✔		✔
drag / drop	✘		✔	✘	✔	✔		✘	✘		✔
widgets	✔		✔	✘	✔	▲		✘	✔		✘
free	✘		✔	✔	✔	✘		✔	✔		✘
reuse	✔							▲	✔	✔	
data struc.	✔							▲	✔	✔	
MVC								✘		✔	
adv. inter.								✔	✔	✔	
portable								✔	✔	✔	
inexp.[a]	$7,000	$$			free	$300+	free	free	free	free	

a. Prices listed in this row are taken from Brad Myers' User Interface Software Tools <http://www.cs.cmu.edu:8001/afs/cs.cmu.edu/user/bam/www/toolnames.html>