

User Interface Specification

Daniel M. Berry

User Interfaces a How Issue?

We used to think that specifying the user interface (UIs) for a system is a How issue rather than a What issue.

That is, it should not be specified in the requirements specification and should be left to the implementers to decide.

However...

However, we have had enough catastrophes in which the culprit was a poor UI interface on the controlling application that left the operator confused as to what was happening and

he or she made a poor choice of what to do.

or

he or she made the wrong request.

We Know Better Now

We have learned that UIs must be considered at requirements time.

The UIs must be carefully designed along with the functional requirements to be consistent with the functional requirements.

Need to Validate UIs

Many times, it is necessary to validate proposed UIs with usability testing with real, alive users.

Finally, the final UIs must be specified in the requirements specification along with all the functional and nonfunctional requirements.

Knew This All Along

Actually, we knew this all along, because ease-of-use is often a nonfunctional requirement, and is thus a requirement that must be specified.

The Harsh Realities

Moreover, we have learned that if U-I issues are not decided upon and specified in the requirements, it often ends up that it is impossible to add them later to the code that results because the proper hooks have not be left in the code.

Even worse than that, it is often necessary to program the function into the UI framework rather than the other way around.

How to Specify User Interfaces

A cool way to specify the UI of a system is to attach screen diagrams to scenario steps.

Doing so has the effect of showing:

- **when a particular screen or window appears,**
- **how the particular screen or window appears, and**
- **what the system does in response to a particular input, including that of selecting or clicking a particular widget.**

To Examples

Let us now specify a reasonable WIMP UI for the *Sensus* system which we have used as an example before.

Recall...

Sensus Modules

Sensus has four main modules:

- ***Registrar — The registrar registers voters prior to an election.***
- ***Pollster — The pollster acts as a voters' [sic] agent, presenting human readable ballots to a voter, collecting the voter's responses to ballot questions, performing cryptographic functions on the voter's behalf, obtaining necessary validations and receipts, and delivering ballots to the ballot box....***

The pollster is the only component of the Sensus system that voters must trust completely; voters concerned about the privacy of their ballots may want to install personal copies of the pollster on trusted machines.

- ***Validator — The validator ensures that only registered voters can vote, and that only one ballot is counted for each registered voter.***
- ***Tallier — The tallier tallies the results of the election or survey. [The word “tallier” should be read as “tally-er”.]***

Registering to Vote

Before registering to vote, a voter must obtain a voter identification number, token, and registration address from the election administrators.

You may begin the registration process by running the pollster module. This is generally done by invoking the sensus command.

The pollster module will display a menu of options. Select the “register to vote” option.

The pollster will generate a public/private key pair for you and then prompt you for your identification number, token, and the registration address.

The pollster will prepare a registration request on your behalf and submit it to the registrar. If all goes well, the pollster will collect an acknowledgment from the registrar within a few seconds. Then, the pollster will prompt you for a file name for saving your registration information. Select a name you will remember, as you will need to tell the pollster

the name of your registration file every time you vote. If you are registered with more than one election authority, make sure you store your registration information in separate files. All Sensus files will be stored in your .sensus directory; if you do not have one, the pollster will create one for you.

Marking Your Ballot

Before you can mark a ballot, you must obtain the unvoted ballot for the election and place it in your .sensus directory. You must also be registered to vote in that election.

Start by running the pollster module as you did when you registered to vote.

If you would like to review the ballot before you mark it, select “view ballot questions and instructions” from the pollster menu.

When you are ready to mark your ballot, select “mark ballot” from the pollster menu.

The pollster will prompt you for the name of the ballot and your registration file name.

The pollster will then display the ballot questions one at a time along with instructions for responding to each question.

[Why not display before, when viewing?]

If you change your mind or make a mistake marking your ballot, you can remark your ballot. At this time it is not possible to change your response to some ballot questions without remarking your entire ballot.

When you have finished marking your ballot, the pollster will prompt you to continue the voting process. By answering yes at each of the prompts, you can authorize the pollster to complete the entire voting process on your behalf immediately. This process usually takes a few minutes. If you do not want to

complete the process right away, you can exit from the pollster program and run it again later to pick up where you left off.

Use Cases

The author of the *Sensus* description has provided us with two use cases, namely for

- registering to vote
- marking a ballot (or voting)

Recall the multi-column, natural language representation of the scenarios of these use cases.

Add UI Specification

At each step in any scenario that a particular screen is desired, we mark the scenario step with a numbered marker and then we show pictures of these screens, each with its own numbered marker.

Each screen picture has a unique numbered marker.

Add UI Spec, Cont'd

Several scenario steps may share the same numbered marker, meaning that the same screen is used for each such step.

Note that in addition to numbering, you can make each numbered marker a hot link, as I did!

UC “Registering to Vote”

Voter	Pollster	Registrar
<p>1. Before registering to vote, a voter must have obtained a voter [identification number, token, and registration address] from the election administrators.</p> <p>2. Voter invokes the <i>sensus</i> command, to run the Pollster.</p> <p>4. Voter selects the “register to vote” option.</p>	<p>3. Pollster displays a menu of options. <i>1</i></p> <p>5. Pollster generates and sends a public/private key pair for Voter. <i>2</i></p> <p>6. Pollster prompts voter for his/her [identification number, token, and registration address]. <i>3</i></p>	

Voter**Pollster****Registrar**

13. Voter sends selected name to Pollster.

14. If Voter has a *.sensus* directory, Pollster creates file with selected name in Voter's *.sensus* directory and stores registration information in the file.

15. Voter exits *sensus* program.

EXCEPTION for step 10:

Voter	Pollster	Registrar
		10. If all does not go well, ...

EXCEPTION for step 14:

Voter	Pollster	Registrar
	14. If Voter does not have a <i>.sensus</i> directory, Pollster creates one and then Pollster creates file with selected name in Voter's <i>.sensus</i> directory and stores registration information in the file.	

ALTERNATIVES for step 7:

Voter	Pollster	Registrar
7. Voter sends his/her [token, identification number, and registration address].		

... one for each permutation of the data...

UC “Marking your Ballot”

Voter	Pollster	Election	Registrar
<p>1. Voter invokes the <i>sensus</i> command, to run the Pollster.</p> <p>3. Voter selects the “marking ballot” option.</p>	<p>2. Pollster displays a menu of options <i>1</i></p> <p>4. Pollster generates and sends a public/private key pair for Voter. <i>2</i></p> <p>5. Pollster prompts voter for his/her [identification number, token, and registration address]. <i>3</i></p>		

Voter	Pollster	Election	Registrar
<p>6. Voter sends his/her [identification number, token, and registration address].</p>	<p>7. Pollster requests verification of voterId from Registrar.</p> <p>9. Pollster requests an unvoted ballot from Election authority to send it to Voter.</p> <p>11. Pollster sends unvoted ballot to Voter. <i>5</i></p>	<p>10. Election authority sends an unvoted ballot to Pollster</p>	<p>8. If voterId is registered, Registrar sends Pollster acknowledgement.</p>

Voter	Pollster	Election	Registrar
<p>13. Voter selects "view ballot questions and instructions" from the pollster menu.</p> <p>15. Voter selects "mark ballot" from the pollster menu.</p> <p>17. Voter sends the name of the ballot and Voter's registration file name.</p>	<p>12. Pollster displays pollster menu. <i>6</i></p> <p>14. Pollster displays ballot questions and instructions. <i>7</i></p> <p>16. Pollster prompts for the name of the ballot and Voter's registration file name. <i>8</i></p>		

Voter	Pollster	Election	Registrar
<p>19. Voter votes on ballot question 1.</p> <p>...</p> <p>21. Voter votes on ballot question m.</p> <p>23. Voter answers "yes" to prompt 1.</p> <p>...</p>	<p>18. Pollster displays ballot question 1 and its voting instructions. <i>9</i></p> <p>20. Pollster displays ballot question m and its voting instructions. <i>10</i></p> <p>22. Pollster prompts Voter with vote-ending prompt 1. <i>11</i></p>		

Voter	Pollster	Election	Registrar
<p data-bbox="135 300 447 369">25. Voter answers "yes" to prompt <i>k</i>.</p> <p data-bbox="135 569 510 602">27. Voter exits <i>sensus</i> program.</p>	<p data-bbox="569 185 933 255">24. Pollster prompts Voter with vote-ending prompt <i>k</i>. <i>11</i></p> <p data-bbox="569 418 956 518">26. Pollster sends completely voted ballot to Election Authority. <i>12</i></p>		

ALTERNATIVE for steps 13–14 for Voter that does not want to review the ballot before marking it:

Voter	Pollster	Election	Registrar
After step 12 comes step 15.			

ALTERNATIVE for steps 18–21 for Voter that quits after some number of ballot questions and goes back to remark the entire ballot, because he or she has changed his or her mind or has made a mistake:

Voter	Pollster	Election	Registrar
After last step among 18–21, go to step 15.			

EXCEPTIONS for steps 22–25:

Voter	Pollster	Election	Registrar
Voter answers “no” to a vote-ending prompt $i < k$ and exits the <i>sensus</i> program, able to continue from the last completed step.			

EXCEPTION for step 8:

Voter	Pollster	Election	Registrar
voterId is not registered ...			

ALTERNATIVE for any Voter step:

Voter	Pollster	Election	Registrar
Voter exits <i>sensus</i> program, able to continue from the last completed step.			

Screen Pictures

The screen pictures follow:

Menu of Options

Screen 1:

Select one option:

Register to vote

Mark ballot

????????

????????

Registration Prompt

Screen 3:

Please enter your

Voter identification no.

Voter token

Registration address

File Name Prompt

Screen 4:

Please enter a file name for saving
your registration information.

OK

CANCEL

Send Unvoted Ballot

Screen 5:

Your pollster has just been sent an unvoted ballot.

You may now choose to mark your ballot.

OK

CANCEL

Pollster Menu

Screen 6:

Select one command:

- View ballot
- Mark ballot
- Resume from where you stopped
- ????????

View Ballot

Screen 7:

Issue/Question/Office 1

Choice 1

Choice 2

•••

Choice j

Vote for
only 1
choice

Issue/Question/Office 2

Choice 1

•
•
•

View Ballot, Scrolled

Screen 7:

⋮

Issue/Question/Office 4

Choice 1

Choice 2

⋮ ...

Choice h

Vote for
at most 3
choices

Issue/Question/Office 5

Choice 1

Name Ballot

Screen 8:

Please enter the name of the ballot for which you wish to vote.

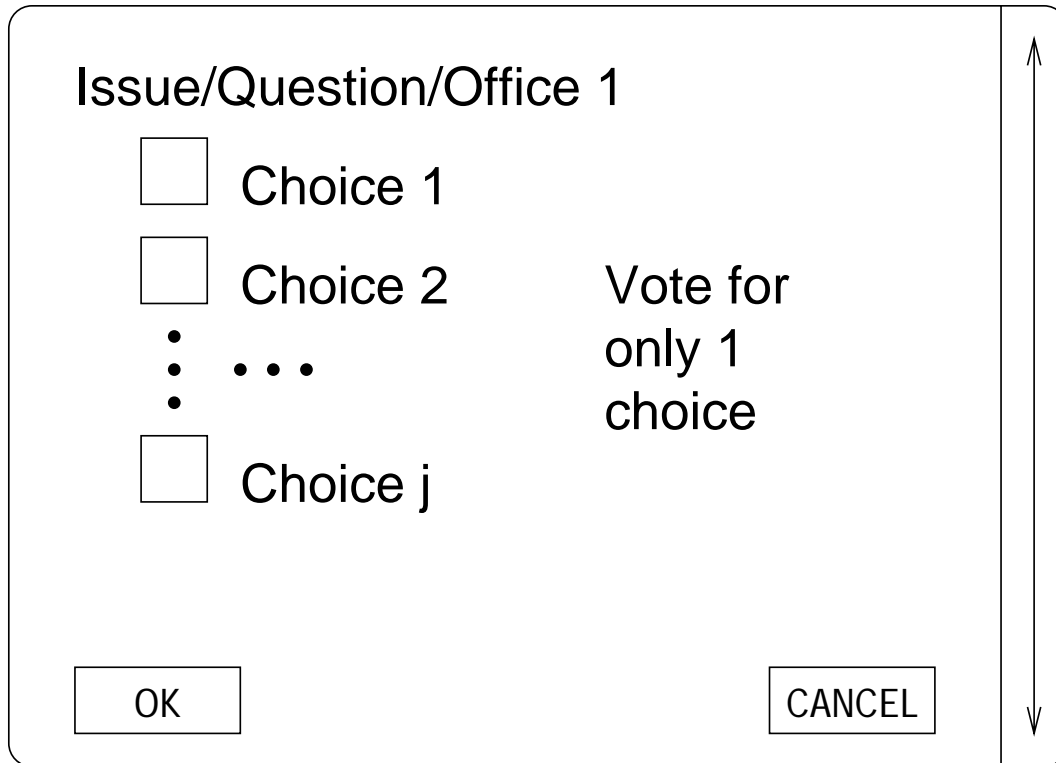
Please enter the name of your registration file, found in your .sensus directory.

OK

CANCEL

Display Ballot Question

Screen 9:



The diagram shows a ballot question screen with a title, a list of choices, a voting instruction, and two buttons. A vertical double-headed arrow on the right side indicates the screen's height.

Issue/Question/Office 1

Choice 1

Choice 2

•••

Choice j

Vote for only 1 choice

OK CANCEL

Display Another Question

Screen *10*:

Issue/Question/Office 4

Choice 1

Choice 2

•••

Choice h

Vote for
at most 3
choices

OK CANCEL

The form is enclosed in a rounded rectangle with a vertical scrollbar on the right side, indicated by an upward-pointing arrow at the top and a downward-pointing arrow at the bottom.

Vote Ending Question

Screen 11:

Vote ending question i

Yes

No

Choose exactly 1 choice

OK

CANCEL

Voted Ballot Sent

Screen 12:

Your voted ballot has been sent to
the election authority.

Answering OK will exit census.

OK

CANCEL

But Whoa!

UCs & Ss are not specifications!

They only *illustrate* functionality from the user's point of view.

Thus a UI specification tied to UCs & Ss is not really a specification.

Tie UI Specs to Specs

For each scenario step with a numbered marker, find the states or transitions in your state machine or process diagrams that correspond to the scenario step, and give the same numbered marker to these states or transitions.

You may need to indicate which transition is taken in response to any textual input or to any widget selection or clicking.

"A riotous book for all us downtrodden computer users,
written in language that we understand."
-Stacy Baratelli, the author's barber

Why Software SUCKS...



and what you
can do
about it

DAVID S. PLATT

Why Software Sucks

In a nutshell,

Software sucks because UIs suck!

For many a program, its UI is not obvious.

The typical non-expert user cannot figure out to get the program to do what he or she wants it to do.

No or Poor Manual

There is often no manual.

When there is a manual, also it sucks, because one cannot easily find the answer to his or her questions about how to use the program.

Poor Help System

There is often a help system, ...

but its index is not very helpful in finding answers to specific questions.

User Feels Stupid

Often, a user is left feeling stupid by his or her inability to get the program to do what he or she wants it to do.

The reality is that the user is fine, but the UI is stupid.

The Way it Ought to Be

A program should be designed in a way that makes consulting a manual or help system unnecessary.

Its UI should guide the user through solutions to his or her problems.

The Way it Usually Is

Instead, the user finds that he or she has to understand the inner workings of the program to use it.

Why Do UIs Suck?

David Platt suggests that the reason a program's UI sucks is that the program's programmers, not professional UI designers, design and implement the program's UI.

The typical programmer programs the interface for a user like him or herself.

The Typical Programmer

The typical programmer wants to be *in control* of all options.

Programming is the *ultimate* expression of this control!

The Typical User

The typical user wants the program to do only what he or she wants; i.e., he or she could not give a s--t about all the options.

How a Program Should Behave

The program should do normally what most people want as a default *without* asking the user to make choices that are probably unintelligible, e.g.,

“Allow / Disallow cookies.”

The program should have an optionally invoked “Preferences” section that guides the user in making his or her choices intelligently, possibly explaining the implications of each choice.

Platt's Law of UI Design

David Platt's First, Last, and Only Law of UI Design:

Know Thy User, for He Is Not Thee.

Control vs. Ease of Use

You, the programmer, may want control.

The typical user wants ease of use.

Example

Directory Assistance:

Control: AT&T' directory assistance just tells you the phone number. You have to dial if you want to.

Ease of use: Verizon's directory assistance tells you the phone number and then dials it for you.

Which choice reflects the way most users operate?

I Don't Care

The typical user says, “I don't care how your program works!” (Actually he or she probably says it with slightly different words!)

Many a programmer forces the user to understand how the program the programmer wrote works in order to use it properly.

Example

The question that the typical text editor or word processor asks when you try to exit the program is, “The text in the file *F* has been changed. Do you want to save the changes?”

Implication of the Question

This question forces the user to understand that the way the program works is that it first reads the file contents into the memory, it modifies the in-memory copy, and then at the end, for the changes to be permanent, the current in-memory must be written back to the file.

A better question is “Do you want to throw away every change you have just done?”

Bad vs Good Feature

On a Windows system, when you select a file *F* and then press the “Delete” key, ...

unless you have figured out how to disable what is about to happen, you get a dialog box that asks, “Are you sure that you want to send *F* to the Recycle Bin?”

Do You Really Want to Be Asked?

When you turn a car's ignition on or off, does the car ask you if you really want to do what you have just done?

The purpose of the recycle bin is to allow recoverable deletes. So why is it necessary to ask if the user is sure that he or she wants a file sent to the recycle bin?

A Better Idea

A better idea is to make as many operations as possible undoable and redoable.

The possible exception would be emptying the recycle bin.

Even that can be made undoable by tying the emptying operation with ???

My Advice

KIS

Keep it Simple!