

Z Specification of User Interface for Application on Mobile Phone

Watcharee Jumpamule, PhD.
Department of Computer Science
Chiangmai University

1

Outlines

- Introduction (Motivation)
- User Interface Design Process
- Formal Specification of User Interfaces
- Z Specification Process
- Example : Tic Tac Toe

2

Introduction

- Careful user interface design is an essential part of the overall software design process.
- The user interface should be design to match the skills, experience and expectations of it's anticipate users.
- Good user interface design can make software easy to understand and use, which results in greater user acceptance.
- In the last couple of years manufacturers have moved away from the more traditional phone interface to more radical layout of buttons and controls.
- Mobile development especially with developing games is a very challenging *due to the fact of several limitations* that mobile handsets naturally inherit.



3

User Interfaces

- User interfaces in general and graphical user interfaces (GUIs) in particular consist of some visual representation of a system and some method for a user to interact with that system.
- the WIMP model (Windows, icons, menus and pointers) is most common in present.
- In terms of an *abstract model* of a system however, all of these methods of interaction can be viewed in the same way. Namely they provide a way for the user to provide input to and receive output from the system.
- The advantages of GUIs include:
 - They are relatively easy to learn and use.
 - The use can use multiple windows for system interaction
 - Fast, full-screen interaction is possible with immediate access to anywhere on the screen.
 - Different types of information can be displayed simultaneously, enabling the user to switch contexts.
 - The user of graphical icons, pull-down menus, buttons, and scrolling techniques reduce the amount of typing.

4

Formal Specification of Graphical User Interfaces

Why Do It?

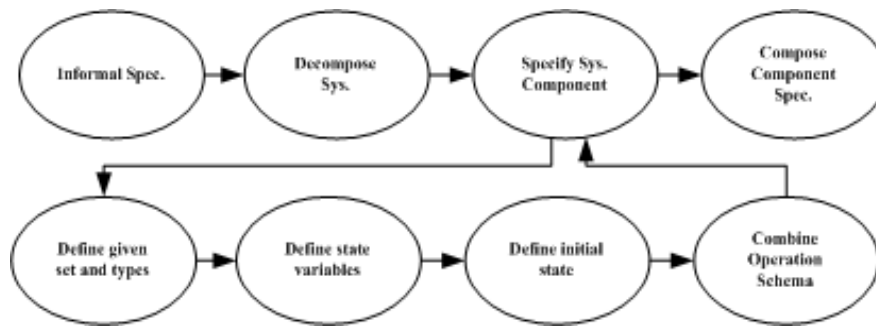
- If we believe that formally specifying a system prior to implementing it is a good thing, which we do, then we can apply the same arguments we use to support this belief to support the inclusion of GUIs into those specification.
- There are many different methods used to provide a description of an interface prior to building it. Whilst these methods can be very successful at providing some initial artefact that can be shared by designers and end-users as a discussion tool, and can be the basis for the implementation of the GUI, *there is no way of formally verifying or proving properties of such models.*
- Problem in using incremental prototypes as the basis for interface design
 - If an early prototype contains an error or problem which is not noticed at the prototype stage then it is possible for this to be built into the final system where it will undoubtedly cause problems at some later stage.
 - If we have a design procedure which allows for a formal model of the underlying system and an informal model for the GUI, then it is virtually impossible to usefully compare the two things or show how they can be linked together until we are at the implementation stage. By then, if there are any problems with the linking it is much harder to fix them. Making changes at this stage is harder and more costly than getting it right at the specification stage. 5

Formal Specification of Graphical User Interfaces

To describe GUIs formally. We identified the following three things which we needed to include in our specification:

1. The system and its operations
2. The behaviour of the interface in terms of its widgets
3. The interaction between the interface and the system

Z Specification Process



1. Write Informal specification of system.
2. Decompose a specification into small pieces called *schemas*.
3. Variables are declared and typed in the top part of *schema*
4. A predicate restraining the possible values of the declared variables is given in the bottom part of the *schema*.
5. Operation expressed by relationship between values of variables before, and values after, the operation.
6. Compose component specification.

Extracted from Sommerville, Ian. **Software Engineering**. 7th ed., New York: Addison Wesley, 2006.⁷

Example : Tic Tac Toe

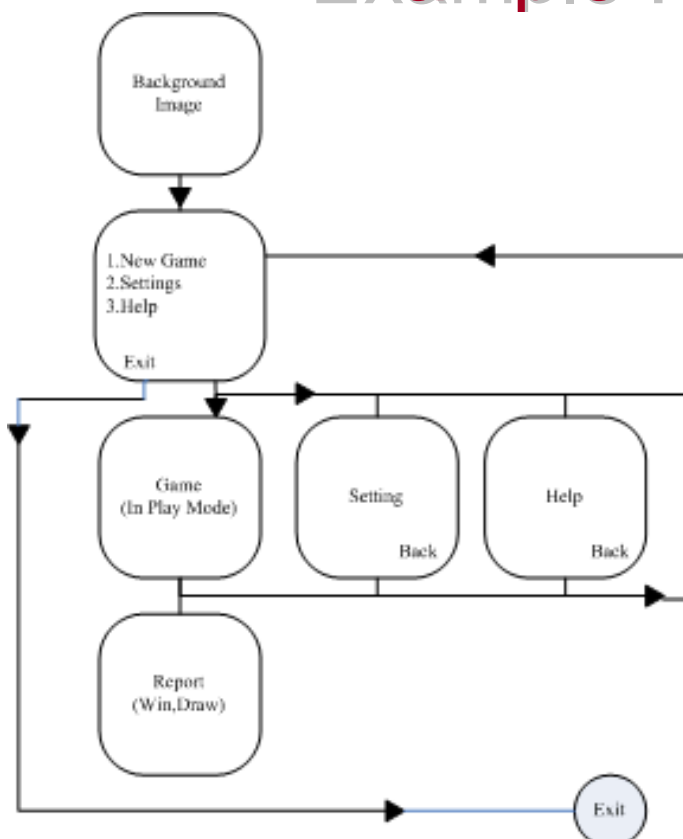
O	X	
	X	
	O	

Informal Description

- The game involves two players and a board. For convenience the players will be called circle and cross.
- The board is a 3 x 3 grid.
- Initially the board is clear.
- The players take turns to move, where a move consists of a player occupying an unoccupied board position, there by adding to their collection of occupied positions.
- The first player whose occupied positions include three positions in a vertical, horizontal or diagonal line (i.e. three in a line) is the winner.

9

Example : Tic Tac Toe



- Suppose that we design menu flow as follows:

10

Z Specification of Tic Tac Toe's User Interfaces

- We may suppose that keys and menu are drawn from basic types:

$[KEY, MENU]$

- We will use the following free type definition to define *Menulist* as a set containing exactly these four values:

$Menulist ::= NewGame \mid Setting \mid Help \mid Quit$

11

Z Specification of Tic Tac Toe's User Interfaces (cont.)

- We use a schema to describe the structure of a Menu:

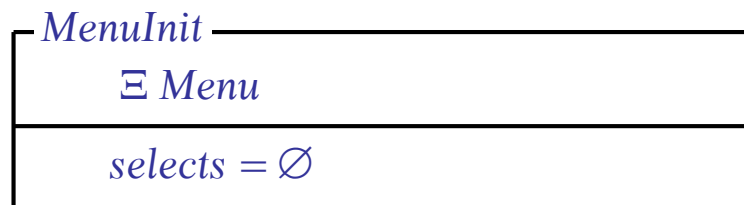
<i>Menu</i>	$selects: \mathbb{P}KEY$ $contents : KEY \rightarrow Menulist$
	$selects = \text{dom } contents$

- A menu should not associate the same key with two different pieces of *Menulist*, hence the requirements that the relation *selects* should be a partial function.
 - *selects* is the set of keys with contents recorded;
 - *contents* is a function, when applied to certain key gives the contents associated with them.

12

Z Specification of Tic Tac Toe's User Interfaces (cont.)

- When a menu is initialized, it selects no data, so the value of *selects* should be the empty function. The following schema describes the initial state of a menu:

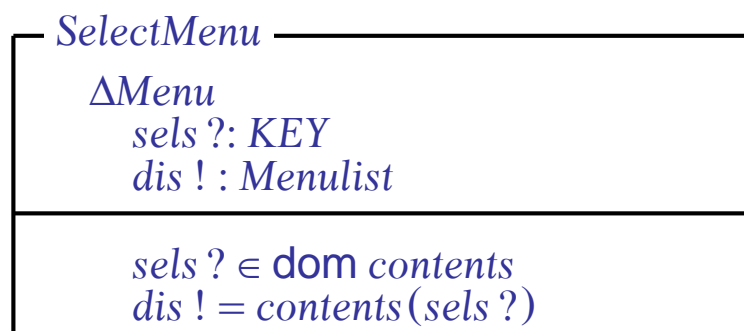


- The predicate says that the set *selects* is empty and a consequence of the *Menu* invariant---the contents function is empty too.

13

Z Specification of Tic Tac Toe's User Interfaces (cont.)

- Then we construct a specification for the operation *SelectMenu*.



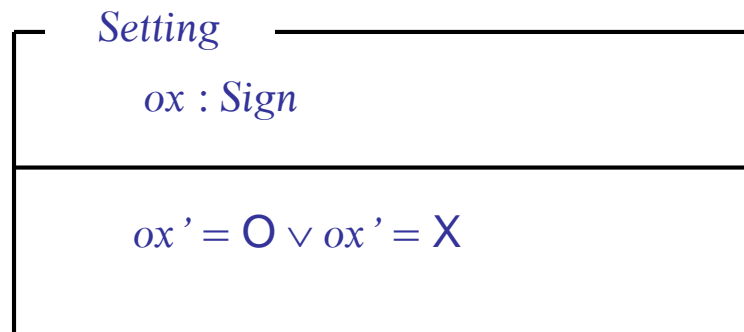
- The precondition for success of this operation is that *sels ?* is one of the keys known to the system, if this is so, the output *dis !* is the value of the *contents* function with an argument *sels ?*.

14

Z Specification of Tic Tac Toe's User Interfaces (cont.)

- We will construct a specification for the *Setting*.

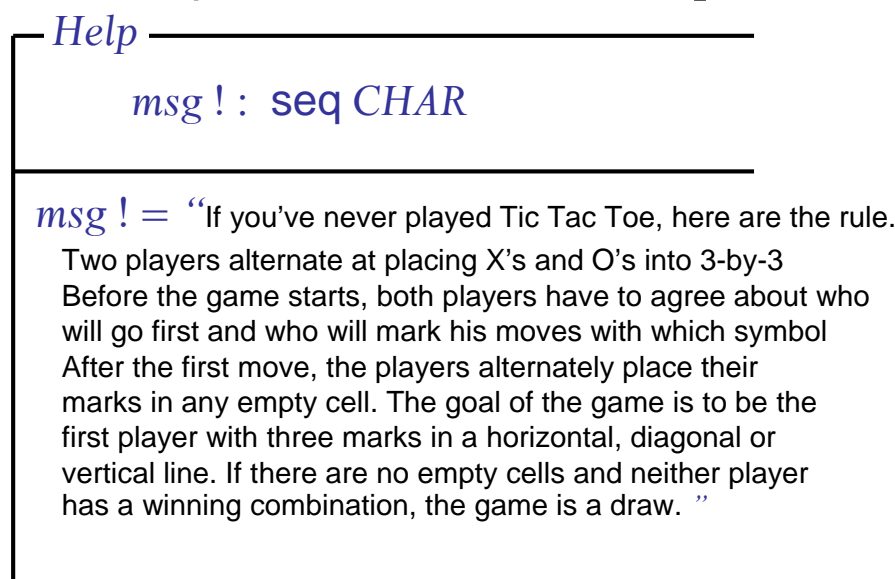
$Sign ::= X \mid O$



15

Z Specification of Tic Tac Toe's User Interfaces (cont.)

- Construct a specification for the *Help*.



16

Z Specification of Tic Tac Toe's User Interfaces (cont.)

- Construct a specification for the *NewGame*.
- We define the set of cells on the game board with *Pos*.

$$Pos == \{1, 2, 3\} \times \{1, 2, 3\}$$

$NewGame$ $\exists Setting$ $Xpos, Opos : Pos$
$Xpos = \emptyset$ $Opos = \emptyset;$ $ox = O \vee ox = X$

17

Z Specification of Tic Tac Toe's User Interfaces (cont.)

- At the start the Tic Tac Toe game, the board is empty, and current player is set to O or X from the Setting.
- We will construct a specification for the *TTTOperation*.
- We define a set called *Mark* to represent the different marked with O or X :

$$Mark ::= O \mid X$$

$TTTOperation$ $\Delta NewGame$ $cell : Pos \leftrightarrow Mark$ $xpos?, opos? : Pos$
$((ox = O) \wedge (cell' = cell \cup \{opos? \mapsto O\})) \vee$ $((ox = X) \wedge (cell' = cell \cup \{xpos? \mapsto X\}))$ $Opos = \text{dom}(cell' \triangleright \{O\}) \wedge Xpos = \text{dom}(cell' \triangleright \{X\})$

18

Z Specification of Tic Tac Toe's User Interfaces (cont.)

O	X	
	X	
	O	

$\{(1,1) \mapsto O, (1,2) \mapsto X, (2,2) \mapsto X, (3,2) \mapsto O, \}$

19

Z Specification of Tic Tac Toe's User Interfaces (cont.)

- We construct a specification for the Report.
- We need a type of output messages:

$Results ::= OWins \mid XWins$

Report

$\exists TTTOperation$

$r! : Results$

$n, m : \mathbb{N}$

$(\exists i: 1..n; \forall j: 1..3; \bullet((i,j) \in Opos \wedge r! = OWins) \vee ((i,j) \in Xpos \wedge r! = XWins)) \vee$
 $(\forall i: 1..3; \exists j: 1..m; \bullet((i,j) \in Opos \wedge r! = OWins) \vee ((i,j) \in Xpos \wedge r! = XWins)) \vee$
 $(\forall i: 1..3; \bullet((i,i) \in Opos \wedge r! = OWins) \vee ((i,i) \in Xpos \wedge r! = XWins)) \vee$
 $(\forall i: 1..3; \forall j: 1..3; \bullet(i+j=4) \wedge ((i,j) \in Opos \wedge r! = OWins) \vee$
 $(i,j) \in Xpos \wedge r! = XWins)$

20

References

- V. Boonnak, and W. Jumpamule. Z Specification of User Interface for Application on Mobile Phone. In *Proceedings of the National Conference in Science and Technology 2006*, Sakon Nakorn, Thailand, December 2006.