Model–View–Controller: TUI Versus GUI

20

The objective of this chapter is to compare and contrast two types of user interfaces and their impact on how the model–view–controller components interact with each other. This chapter also introduces some basic components found in a graphical user interface.

20.1 Preconditions

The following should be true prior to starting this chapter.

- You understand the four criteria used to evaluate a HCI design:

  1. Efficiency—How does the HCI affect the productivity of the user?
  2. Learnability—How easy is it for a user to learn the HCI?
  3. User satisfaction—How satisfied is the user with the HCI?
  4. Utility—Does the HCI provide useful and timely information?

- You understand the HCI design goals: know the user, prevent user errors, optimize user abilities, and be consistent.
- You appreciate the value user participation brings to the HCI design process.
- You understand the six characteristics of a good software design: simplicity, coupling, cohesion, information hiding, performance, and security.
- You understand the notion of abstraction as a design process. Abstraction is the ability to generalize a concept by removing details that are not needed to properly convey the design perspective being emphasized.

D. P. Voorhees, *Guide to Efficient Software Design*, Texts in Computer Science,
https://doi.org/10.1007/978-3-030-28501-2_20

- You have created and/or modified models that described an object-oriented or structured software design.
- You understand how to apply the Model–View–Controller architectural pattern to modify an existing design into these three components.
- You understand how to apply the Model–View–Controller architectural pattern to create a high-level design that accurately reflects the domain requirements while satisfying the design constraints inherent in MVC.

20.2 MVC User Interface: Concepts and Context

When using the MVC architectural pattern, the controller and view components must interact with each other to support the application domain and user interface. The details of how these two components work together to support the application is largely dependent on whether the user interface is implemented as a text-based or graphical-based user interface.

20.2.1 Text-Based User Interface

Prior to Chap. 17, the ABA case study provided a text-based user interface (TUI) where the domain logic implemented in the controller enforced a specific sequence of user interactions. The user first had to enter a contact name or the word "exit". If "exit" was entered, the contents of the address book would be displayed and the application would end. Otherwise, the user would next enter a phone number and then an email address. Once these three data values were entered, in this specific order, they would be validated and either added to the address book or error messages would be displayed to the user.

In Chaps. 18 and 19, the ABA case study was modified to first display a menu. This gives the user the option of entering new contact data, displaying data for a person already stored in the address book, or exiting the application. When the user decides to create a new contact, the user must enter the three data values—name, phone number, and email address—*in this specific order*.

20.2.2 Graphical-Based User Interface

With a graphical user interface (GUI), the user controls what they do next. A GUI for the ABA case study will allow the user to decide whether to create a new contact, display data for an existing contact, or exit the application. In addition, when the user decides to create a new contact, a GUI design will likely allow a user to enter the three data values—name, phone number, and email address—*in any order they prefer*.

The other aspect of designing a GUI is the types of user controls a designer may use to enable user interactions. The list below identifies some of the common user interactions and the types of user controls used to support the interaction type.

User requests an action: A menu may identify various actions (e.g., File-Open, File-Save As), a push button represents a distinct action (e.g., Save, Cancel), or a hypertext link may initiate a particular action (e.g., show a different web page).

User selects a choice: A list box is typically used to display a list of items from which one may be selected, a small group of option buttons (also known as radio buttons) allows the user to select one option from the group, or a drop-down list allows the user to display the list to select an item.

User selects many choices: A list box may be used to allow the user to select multiple items, or a small group of check boxes allows the user to select as many options as they desire.

User enters data: A text box allows the user to enter a single line of text, a multi-line text box supports entry of large amounts of text, or a combo box (i.e., a text box that includes a drop-down list) allows a user to select from the drop-down list or enter data in the text box to find a matching item in the list.

20.2.3 Summary: TUI versus GUI

A text-based user interface requires the software to guide the user through the steps needed to complete a processing step. In terms of data entry, a TUI forces a user to enter data in a predetermined order. A graphical user interface allows the software to provide actions from which the user may select, and allows data entry to be done by the user (typically) in any order.

20.2.4 Impact on MVC Design

With a TUI, the controller component will call methods within the view component. These method calls allow the software to control choices the user has within the user interface. This includes sequencing the entry of data. Figure 20.1 shows a UML communication diagram for a generic text-based user interface. The controller component would call a method/function in the view to start the user interactions. The specific user interactions could be done within the view. As shown in this figure, the view would need to obtain user data in a specific sequence, then return the data as part of the generic method/function call getUserData.

With a GUI, the view component displays a window/page, the user chooses what to do based on the choices presented, and then the user initiates an action (via the GUI) that requires the view to communicate with the controller. Figure 20.2 shows a UML communication diagram for a generic graphical user interface. The view component is responsible for displaying the GUI and reacting when the user requests a specific action. Typically, the view component would create an object/data entity that is then

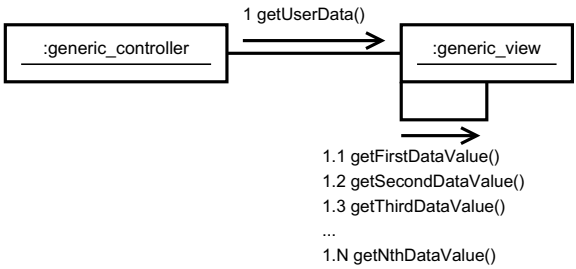

Fig. 20.1 Generic communication diagram—Text-based user interface

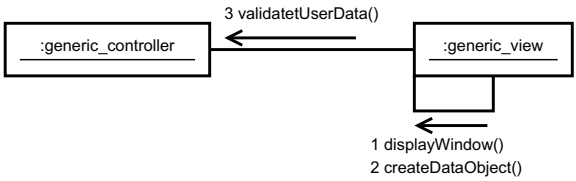

Fig. 20.2 Generic communication diagram—Graphical-based user interface

sent to the controller. This object/data entity would contain the data entered by the user and also identify the type of action the user requested. The controller would validate the data entered and decide what to do based on the user requested action and validation results.

20.2.5 MVC Summary: TUI Versus GUI

With a TUI, the controller controls the user interactions by calling view methods. The view will provide a particular sequence that the user must follow when using the application. With a GUI, the view controls the user interactions. The view will provide a graphical allowing the user to make choices about what to do next. The view will call a controller method to inform the controller what the user has done/requested.

20.3 GUI: More Concepts and Context

A graphical user interface (GUI) is the most common type of HCI in use in modern software. This includes support for touch screen devices (e.g., *smart* cell phones), web pages, and software applications that operate within desktop/laptop environments. In this chapter, we'll focus on GUI designs for desktop/laptop software applications.

A GUI design must integrate three different types of software elements.

1. Graphical components or objects.
   These components include GUI objects that contain other GUI objects (i.e., container objects) as well as basic GUI objects representing a distinct type of user

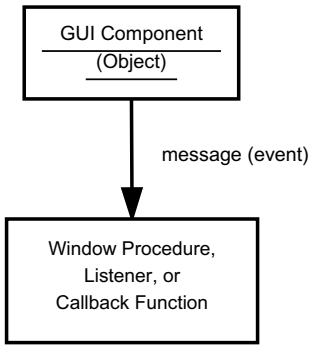

Fig. 20.3 Relationship between GUI objects and Listeners/Callback functions

interaction. The container GUI objects include windows, dialog boxes, and group boxes. The basic GUI objects include labels, buttons, text fields, menus, and list boxes. There are also GUI objects that combine two more basic objects. An example of this is a drop-down combo box, which combines a text box with a list box.

2. User events or messages.
   An event is an action, often a user action, that is responded to by the software. Examples of events are: a mouse button is clicked; a keyboard key is pressed; a graphical button object is clicked; a mouse pointer is moved; a mouse pointer is clicked-and-dragged; and a timer expires. A message is the mechanism used to convey the event to the application code.
3. Code that reacts to user events and manipulates the graphical components/objects. The code that reacts to events is known as window procedures (in C++), window listeners (in Java), and callback functions (in Python). Figure 20.3 shows a GUI object generating a message that is sent to the Window procedure, listener, or callback function. This message contains the event that has been triggered based on (typically) a user action on a GUI object. A window procedure, listener, or callback function *is idle* until an event triggers this logic to be executed. The logic we put in a procedure/listener/function responds to the event in an appropriate manner. This may include ignoring the event.

20.3.1 GUI Design Alternatives

A HCI designer will generally identify the types of GUI components needed and how these components will appear within a window container. The designer will also identify the (user) events that will need to be responded to. When multiple windows are needed, the designer must also determine which actions will cause another window to be displayed.

The commonly used GUI components and their associated events are predefined by a programming language library. For C++, you'll find GUI objects and events

in the Windows 32 API and in the Microsoft Foundation Classes. For Java, you'll find GUI objects and events in the Java API class hierarchy, specifically the java.awt (abstract windowing toolkit), javax.swing, and javafx packages. In Python, you'll find GUI objects and events in the graphical user interfaces with Tk/Tcl, specifically tkinter and related modules.

Some of the commonly used GUI components are now described, in alphabetical order.

20.3.1.1 Check Box

A check box is typically displayed as a small square with a description to the right of the square. A check box represents an option among a group of independent options. An example is the options (e.g., floor mats, CD player, power locks, power windows) when buying a new vehicle.

C++ Win32: Use the BS_CHECKBOX style of the Button class.
Java: Use the JCheckBox class.
Python: Use the tkinter.CheckButton class.

20.3.1.2 Label

A label is typically displayed as text with no borders. It represents static text (i.e., text that cannot be modified by the user) displayed in the window.

C++ Win32: Use the Static class.
Java: Use the JLabel class.
Python: Use the tkinter.Label class.

20.3.1.3 List Box

A list box is typically displayed within a rectangle with borders and a vertical scroll bar. It represents either a list of mutually exclusive options (like a group of radio buttons) or a list of independent options (like a group of check boxes).

C++ Win32: Use the Listbox class.
Java: Use the JList class.
Python: Use the tkinter.Listbox class.

20.3.1.4 Push Button

A push button is typically displayed as a rounded rectangle with a verb or verb phrase inside the rectangle. A push button denotes an action that will be performed when the button is activated by the user. Examples of push buttons are send (in an email application) and save (in a word processing application).

C++ Win32: Use the BS_PUSHBUTTON style of the Button class.
Java: Use the JButton class.
Python: Use the tkinter.Button class.

20.3.1.5 Radio Button

A radio button is typically displayed as a small circle with a description to the right of the circle. A radio button represents one option among a group of mutually exclusive options. An example is a list of color options for a new vehicle. Usually, a group of radio buttons is defined so only one can be selected at a given point in time.

C++ Win32: Use the BS_RADIOBUTTON style of the Button class.
Java: Use the JRadioButton class. Put these inside a ButtonGroup object to identify the group of buttons requiring mutually exclusive selection.
Python: Use the tkinter.Radiobutton class. Associate each Radiobutton object with the same variable to group the buttons requiring mutually exclusive selection.

20.3.1.6 Text Fields

A text field is typically displayed as a rectangle. It represents a place where a user can enter and modify text data found within the text field. Some text fields allow one line of text to be entered while another type of text field allows the user to enter multiple lines of text.

C++ Win32: Use the Edit class.
Java: Use the JTextField or JTextArea class for a single or multi-line text field, respectively.
Python: Use the tkinter.Entry class.

20.4 Post-conditions

The following should have been learned when completing this chapter.

- Using Model–View–Controller for a text-based user interface means the application controls the sequence of user interactions. This is done by having the controller component call view methods/functions.
- Using Model–View–Controller for a graphical user interface means the user controls the sequence of interactions. This is done by having the view component call controller methods/functions.

Exercises

Discussion Questions

1. What types of applications may benefit from having a text-based user interface?
2. What arguments could you make for an MVC TUI design behaving more like a GUI when the user enters commands?