Development of Graphical User Interface by Applying Philosophy of Use Case Maps

Ebitisam K. ELBERKAWI *, Mohamed M. ELAMMARI **

* Academy of Graduate Studies, Benghazi, Libya
Bit2gar@gmail.com
**Faculty of Information Technology, Garyounis University, Benghazi, Libya
Elammari@garyounis.edu

Abstract: *Graphical User Interface (GUI) development is as important as other aspects and phases of programming, as GUI elements are typically the only means by which the user interacts with the program. Thus, even if the functionality of the application is perfectly executed, if its GUI is difficult to use, this can result in an implementation failure. In such cases, the user will switch to another application, which is easier and more flexible, even if its internal functionality is inferior. Therefore, the GUI must primarily be user-friendly, and must be built on strong foundations and pillars. This paper introduces an approach for designing GUI based on a model that offers visual description of high-level system logic—Use Case Maps (UCMs), which is utilized in three phases.*

Keywords: *Graphical User Interface, Use Case Maps.*

1. Introduction

There is no doubt that the role played by the Graphical User Interface (GUI) in making systems more successful is very important. In practice, it is not uncommon for strong performance systems to fail at their implementation stage due to the shortcomings in their GUI. The GUI is the interface between the user and the system and thus plays the key role in the application acceptance. In short, although GUIs typically include less complex elements to develop, they are critical to the success of the system. Moreover, given that their acceptance relies on user preference, they are often the hardest part of the system to develop. Thus, GUI designers and programmers must achieve a high degree of cooperation in the design of GUI. Analysis and design phases are integral part of project life cycle and they must incorporate GUI from the inception. In addition, user input is essential aspect of GUI construction and design. This paper presents an approach to GUI design that is based on the primary phases of system development and supports the role of the user. Such approach increases the chance of cooperation between the GUI designer and programmer as well as ensures that the final product meets user requirements.

In this paper, we focus on the design of GUI through Use Case Maps (UCMs) as one of high-level models. The approach proposed in this paper is composed of three phases, starting from UCMs and ending in GUI: the first phase (Responsibility Analysis) produces a Responsibility Diagram (RD); second phase (GUI Design) produces GUI Design Tables (GUI_DT), which contain all details of designing the GUI; and last phase (GUI Development) produces the GUI. The rest of the paper is organized as follows: Section 2 gives a summary of some pertinent literature, where key findings and comparisons of these works are discussed. Section 3 describes the three phases of our GUI design approach in more detail. Its implementation was further elaborated on through a case study in Section 4. Finally, Section 5 presents discussion, followed by the key conclusions and recommendation for future works.

2. Literature Review

Most of the extant studies related to our work focus on achieving GUI design using Unified Modeling Language (UML), as was done in works [4,5,7]. There is also some evidence [4,7] of a GUI design through Use Cases. Thus, it is worth to briefly elaborate on the role of Use Cases and system details they present, which is at the level of the system functions from viewpoint of external actors. On the other hand, work by Almendros-Jim´enez and Iribarne [7], focus has been made on the relation <include> without knowing the role of the relation <extends>, as each of these two relations has an important role in Use Case Model. Moreover, Pinheiro and Paton [5] focus on UI modeling by means of UML. Within this work, we have seen strong and weak aspects in the UML and have identified their respective roles in UI modeling.

Same authors in [6] primarily focused on the aspects of GUI that are not covered by UML. Hence, they introduced additional notation when designing Interactive Applications. Thus all previous attempts to design a GUI have both advantages and disadvantages and, in general, most depend on Use Cases. Thus, the aim of this work is to design a general approach through which GUI is developed, using UCMs as a high-level model .

The main advantage for selecting UCM for the design of GUI is based on its many advantages, including high-level design view, combining structure and behavior in one view in a graphical, and easy to understand way, in addition to its compact form, which provides wealth of information [1, 2].

Moreover, an important and useful linkage between UCM and many kinds of diagrams for one of the most known modeling languages, namely UML, can be established. UCM acts as an intermediary between Use case diagrams (UCDs), which describe the system without any details (Black-Box), and other behavioral diagrams, which describe more details that make understanding the system very difficult (Glass-Box). Moreover, UCM provides a link between behavioral diagrams and structural diagram. In this respect, UCM can be perceived as Gray-Box [3], as it bridges the gap between requirements and design. Based on aforementioned features and support for the competence of UCM, it has been selected to be the basis from which the GUI will be developed.

UCMs are precise structural entities that contain enough information in highly condensed form to enable a person to visualize system behavior. UCMs (as shown in Figure 1) provide a high-level view of causal sequences in the system as a whole, in the form of paths, which are known as scenarios. In general, UCMs may have many paths (for simplicity, the figure only shows one). Although the causality expressed by the paths is understood by humans, this may not necessarily be the case for individual system components.

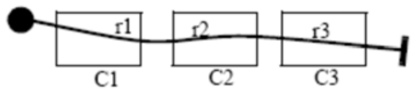


Figure 1: Example of a UCM.

A filled circle indicates a start-point of a path—the point where stimuli occur, causing activity to start progressing along the path. Similarly, a bar indicates an end-point—the point where the effect of stimuli are felt. Thus, paths trace causal sequences between start- and end-points. The causal sequences connect responsibilities, indicated by named points along paths (e.g., r1, r2 and r3). Paths are superimposed on rectangular boxes representing operational components of the system (e.g., C1, C2 and C3), to indicate where components participate in the causal sequences. Individual paths may cross many components and components may have many paths crossing them.

The basic assumption is that stimulus-response behavior can be represented in a simple way with paths. This is a very common characteristic of the types of systems with which we are concerned and results in a path-centric system view, rather than a conventional component-centric view.

3. The Approach

The approach proposed in this paper comprises three phases. In the first phase, rules are applied to UCMs, where the RD will be generated. The second phase involves applying rules to RD, where we will have contents of GUI_DT in their different kinds. In the last phase, contents of the GUI_DT are converted into their corresponding GUI elements, resulting in a complete GUI with all of its elements. Thus, the elements of GUI covered in our work are Form, Button, Textbox, Label, and List. Figure 2 and Table 1 clarify the approach phases and their relationship, as well as the professional in charge for the implementation of each phase.

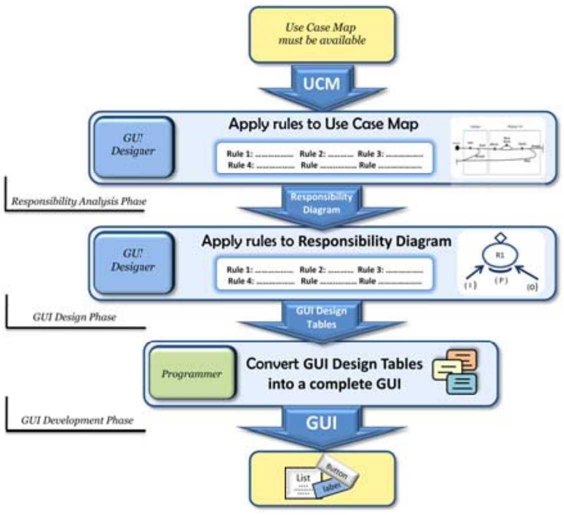


Figure 2: The Approach Phases.

Table 1: The Approach Phases.

Phase Name	Availability of	Description	Result
Responsibility Analysis	UCM Of the system	Apply rules to UCM	RD
GUI Design	RD	Apply rules to RD	GUI_ DT
GUI Development	GUI_ DT	Convert tables into design of GUI	GUI

3.1 First Phase: Responsibility Analysis

3.1.1 Rules

A number of rules have been proposed, depending on the importance of input, processing, output, and interaction with the user for each responsibility of the UCM. In this context, the responsibility refers to tasks, actions or functions to be performed. This distribution of responsibilities was the basis of construction of most rules presented below, with the aim to achieve the desired results through application.

Rule 1
A component in UCM is a form, where this component contains at least one responsibility that requires GUI elements.

Rule 2
Sequence of forms is based on the sequence of components in UCM.

Rule 3
A responsibility will be considered if it requires GUI elements for:

- Obtaining required input for implementing its work
- Executing its work
- Showing the output resulting from implementing its work
- Interacting with the user to show him a message

Rule 4
A responsibility that needs GUI elements for its input must have:

- (X) text boxes
- (X) labels
- (2)buttons (‘Ok’ and ‘Cancel’ button)

where X is a number dependent on the number of data inputs.

Rule 5
A responsibility that requires GUI elements for executing its work must have:

- (1) button (Execute Button)

Rule 6
A responsibility that requires GUI elements to display its output must have:

- (Y) lists
- (Y) labels

where Y is a number dependent on the number of resulting outputs.

Rule 7
A responsibility that needs GUI elements for interaction with the user is a separate form containing a message with a button the function of which is to close the display form.

Rule 8
If a responsibility requires a sub-input based on essential input (Multi-input), then, an independent form is generated to receive additional input using the same input rules as defined in Rule 4.

Rule 9
If a responsibility requires a sub-output based on essential output (Multi-output), then, an independent form is generated to show additional output using the same output rules as described in Rule 6.

Rule 10
The main form of the system is generated based on forms obtained from above-mentioned rules, i.e. using (Bottom_Up) approach.

Rule 11
For each main form, there is an ‘Exit’ button and a ‘Return’ button for each sub-form.

Rule 12
Applications of specific graphical user interface [13], such as ATM system, require no main form and no ‘Exit’ and ‘Return’ buttons. This type of systems has relay forms with no need to return to the main form to select a specific form. Accordingly, a ‘Welcome’ form is displayed in the beginning of user interaction, which is followed by the remaining forms.

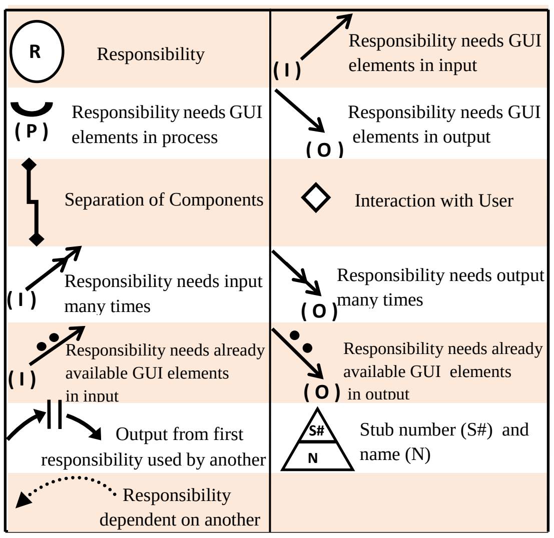
Rule 13
Cases that require more than one scenario represented by static stub notation using (Or) relation are translated into a number of buttons equal to the number of scenarios, where each button may cause moving to the form of the chosen scenario.

3.2 Second Phase: GUI Design

3.2.1 Responsibility Diagram (RD)

RD is a diagram the notation of which has been built and especially designed to show what may result from the process of applying rules to UCMs. Table 2 explains RD notation.

Table 2: RD Notation.



R Responsibility	(I) Responsibility needs GUI elements in input
(P) Responsibility needs GUI elements in process	(O) Responsibility needs GUI elements in output
Separation of Components	Interaction with User
(I) Responsibility needs input many times	(O) Responsibility needs output many times
(I) Responsibility needs already available GUI elements in input	(O) Responsibility needs already available GUI elements in output
Output from first responsibility used by another	S# N Stub number (S#) and name (N)
Responsibility dependent on another	

3.3 Third Phase: GUI Development

3.3.1 GUI Design Tables (GUI_DT)

As a result of the previous phase, i.e. applying rules to RD diagram, all necessary information about required GUI is obtained and is sorted into tables called GUI_DT. These tables are classified into four types, each providing information on the aspect of the GUI design required. Table 3 provides a list of these four tables and the corresponding explanation of their general design. The four tables presented are:

1. Components State Table
2. GUI Elements Tables
3. Prerequisites Tables
4. Interactions Tables

Table 3: General Design of GUI_DT.

1. General Design of Component State Table
Number: One table for each system.   **Explanation:**   In this table, there is an illustration for each component of the system UCM, required to design a GUI, in terms of whether this component has or does not have the following: • GUI elements in accordance with the contents of this component of responsibilities that require them.   • Interaction with the user by showing him/her a message based on the responsibilities that require interaction.   **The meaning of symbols and figures**   **Symbols**   X Indicates possession.   - Indicates no possession.   **Figures**   None

2. General Design of GUI Elements Tables
Number   A number of tables exceeding or equal to the number of components that have GUI elements described by Components State Table.   **Explanation**   In this table, there is an illustration of each responsibility that requires GUI elements, in terms of input, output and process, and the type and number of these elements. Note that the table containing the responsibility of the type that requires multi-input or output will have a one or more sub-tables, depending on the requirements of this responsibility.   **Note:** Using these tables, the necessary clarification of static stub can be determined, once it exists in UCM of the studied system.   **The meaning of symbols and figures**   **Symbols**   None   **Figures**   Figures of this type of tables determine the number of each element of the GUI and there is a total number of all totals in the last row of the tables.

3. General Design of Prerequisites Tables
Number   A number of tables equals to GUI Elements Tables.   Number of Prerequisite tables = Number of GUI elements tables   **Explanation**   In these tables, the prerequisites that must be met for each responsibility that requires GUI elements is clarified, in terms of whether the prerequisite is relative to another responsibility and its location, as well as the component and the type of this relation.   **The meaning of symbols and figures**   **Symbols** $\uparrow$ Relative to responsibility in previous component $\leftrightarrow$ Relative to responsibility in the same component   **Figures**   None

4. General Design of Interactions Tables
Number   A number of tables equals to the number of components that interact with the user and are thus described by Component State Table.   Therefore, the number of these tables is equal to or less than the number of components of the system that GUI is designed for.   Number of interaction tables <= Number of components   **Explanation**   These tables clarify each responsibility that needs to interact with the user.   **The meaning of symbols and figures**   **Symbols**   None   **Figures**   None

4. Case Study

For clarification, three different study cases have been practically applied. This diversity was useful when applying all derived rules and verifying their suitability for different systems that require GUI. For brevity, however, only one of case study (Online shopping) is presented here. In this context, online shopping is defined as an entire shopping process conducted over the Internet, the popularity of which is increasing steadily [8]. The implementation of our work will be on a specific part of the system described in Figure 3 [9, 10, 11].

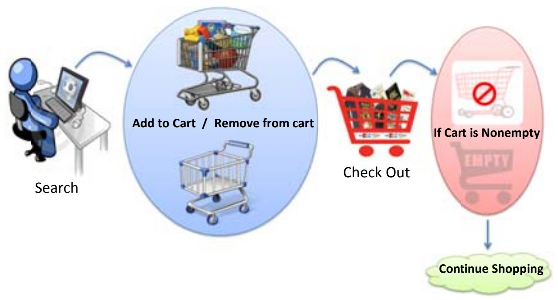


Figure 3: Online Shopping System.

4.1 Practical Application of First Phase (Responsibility Analysis)

In this phase, as already explained, a UCM must be provided for the system for which we need to develop a GUI by applying rules to UCM to obtain an RD. Figure 4 shows a UCM of Online Shopping System.

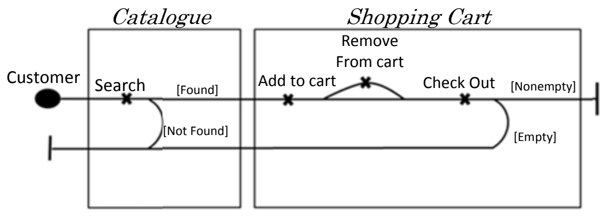


Figure 4: UCM of online shopping system.

At the end of this phase, and applying the rules to UCM of Online Shopping System, an RD is obtained, as shown in Figure 5, using the notation already clarified.

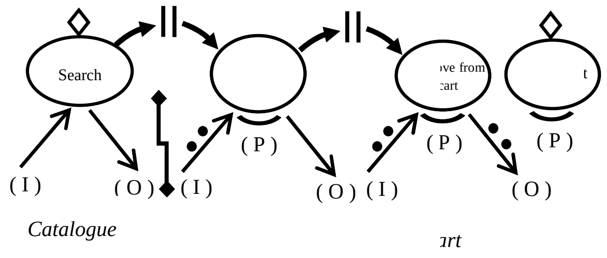


Figure 5: RD of Online Shopping System.

4.2 Practical Application of Second Phase (GUI Design)

In this phase, rules are applied to RD as a result of previous phase, yielding data for GUI_DT. At the end of this phase, the GUI_DT (shown in Table 4) will yield four table types, as described above.

Table 4: GUI_DT of online shopping system.

I. Components State Table

Component Name	GUI Elements	Interact with User
Catalogue	x	x
Shopping Cart	x	x

II. GUI Elements Tables

II.1 Component name : Catalogue

Graphical Element		Label	Button		Text	List	Description
Responsibility	Type						
Search	I	1	2	• Ok, Cancel	1	-	Search
	O	1	-	-	-	1	Results
Total		2	2		1	1	6

II.2 Component name : Shopping Cart

Graphical Element		Label	Button		Text	List	Description
Responsibility	Type						
Add to cart	P	-	1	• Execute Button	-	-	Add to cart
	O	1	-	-	-	1	Shopping cart
Remove from cart	P	-	1	• Execute Button	-	-	Remove from cart
Check Out	P	-	1	• Execute Button	-	-	Check Out

Total	**1**	**3**	**-**	**1**	**5**

III. Prerequisites Tables

III.1 Component name : Catalogue

Responsibility	Prerequisite	Type of Prerequisite	Suggesting
Search	-	-	-

III.2 Component name : Shopping Cart

Responsibility	Prerequisite	Type of Prerequisite			Suggesting
Add to cart	Availability of items that can be added to cart	Relation with another Responsibility			As a result of such link to another responsibility in another component by the output, a link must be provided between these two forms (Thus, we propose merging both forms into one)
		Responsibility name	***Type of relation***	***Location***	
		Search	Output	↑	
Remove from cart	Cart must be nonempty (Items added to cart)	Relation with another Responsibility			-
		Responsibility name	***Type of relation***	***Location***	
		Add to cart	Output	↔	
Check Out	-	-			-

IV. Interactions Tables

IV.1 Component name : Catalogue

Responsibility	Description Of Interact
Search	If item is not found as a result of searching

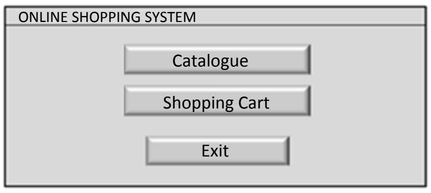
IV.2 Component name : Shopping Cart

Responsibility	Description Of Interact
Check Out	The user cannot continue because shopping cart is empty

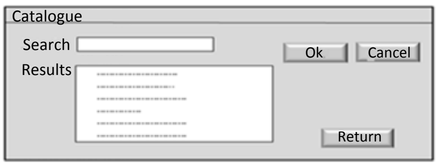
4.3 Practical Application of Third Phase (GUI Development)

The above GUI_DT provide all information needed to design the GUI and its elements. The design shown below (Figure 6).

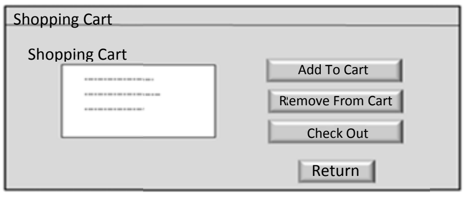
I. Main Form



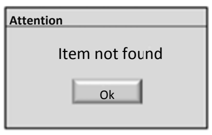
II. Catalogue

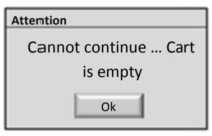


III. Shopping Cart Form



IV. Interaction Forms





According to the suggestion mentioned in Prerequisites Tables, the two forms will be merged into the following form. This proposal is put forward by GUI Designer to the programmer.

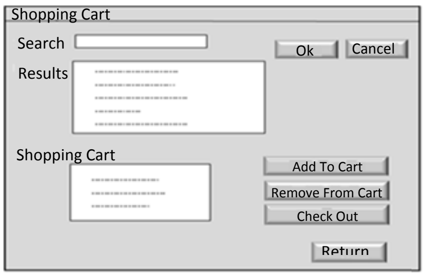


Figure 6: Design of GUI of Online Shopping System.

5. Conclusions and Recommendations

5.1 Conclusions

Given the crucial role of GUI in successful system design and application, as the only means of interaction between the system and the user, we have proposed an approach to develop GUI from UCMs. This approach comprises three consecutive phases—Responsibility Analysis phase, GUI Design phase, and GUI Development phase. Each phase is conducted according to the rules that guide the implementation of this approach, the first and the second phase in particular. By obtaining GUI design for the system of the case study and comparing it with the presented scenario, we confirmed that it satisfied the needs of this system in terms of GUI elements gained through UCM—the basic foundation of our work. This enabled us to design a GUI through primary phases of analyzing and designing systems, thus our paper contributes to the enrichment of this important aspect i.e. the role of analysis and design of systems in GUI design and development. Moreover, the approach presented here provides objective and easy to use tools that stimulate cooperation between GUI designer and the programmer and contribute in building GUI application components, whilst incorporating not only design and programming views, but also the user requirements.

5.2 Future Studies

We recognize that our work was delimited to a small range of applications and identify several issues that may be considered in future studies.

First: In our work, we dealt with most of the basic notation of UCM. On the other hand, we mentioned Static stub as an advanced classified notation of UCM. Therefore, looking at the rest of the advanced notation, especially Dynamic stub, to know its role in designing GUI will be very valuable in studying these issues further on.

Second: There are many issues related to GUI elements that could be a subject of future studies, such as Radio Button and Check Box, to name a few. Their inclusion into the GUI design would result in marked improvements that could increase the scope of applicability of the resulting GUI.

Finally: The approach presented here should be tested on different systems in order to identify any potential issue and work towards resolving them.

References

[1] Buhr, R J A and Casselman R S, *Use Case Maps for Object-Oriented Systems*, Prentice Hall, 1996.

[2] Amyot, D. and Mussbacher, G., "On the Extension of UML with Use Case Maps Concepts," <<UML>>2000, 3rd International Conference on the Unified Modeling Language, LNCS 1939, 16-31, York, UK, 2000.

[3] Amyot, D, Use Case Maps and UML for Complex Software-Driven Systems, Unpublished, Online at http://www.usecasemaps.org/pub/uml99.pdf 1999.

[4] Elkoutbi M. and Keller R. K., "User Interface Prototyping Based on UML Scenarios and High-Level Petri Nets," In 21st International Conference on Application and Theory of Petri Nets (ICATPN '00), pages 166-186, 2000.

[5] Pinheiro da Silva P. and Paton N.W., "User Interface Modeling with UML," Information Modeling and Knowledge Bases XII, H. Jaakkola, H. Kangassalo, and E. Kawaguchi, eds., IOS Press, pp. 203–217, 2001.

[6] Pinheiro da Silva P., Paton N., "UML*i:* The Unified Modeling Language for Interactive Applications," In Proceedings «UML» 2000, Evans, A., Kent, S. (Eds), LNCS, Vol. 1939, 117-132, Springer-Verlag, 2000.

[7] Almendros-Jim´enez J. M. and Iribarne L., "Designing GUI Components for UML Use Cases," In 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05), pages 210–217, IEEE Computer Society Press, 2005.

Web Sites

[8] Http://www.earlyimpact.com/productcart/shopping-cart-software-101.asp.(07/2010)

[9] Http://www.eshoppingindia.com/info/online-shopping-guide.html.(07/2010)

[10] Http://www.indiaplaza.in/quickstep.aspx.(06/2010)

[11] Http://www.tesco.ie/easySteps.htm. (07/2010)