MODEL-BASED USER INTERFACE DESIGN

A Project

Presented to the faculty of the Department of Computer Science
California State University, Sacramento

Submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE
in
Software Engineering

by
Vishwa Kishore Mannem

SPRING
2016
MODEL-BASED USER INTERFACE DESIGN

A Project

by

Vishwa Kishore Mannem

Approved by:

______________________________, Committee Chair
Ahmed Salem

______________________________, Second Reader
Robert Buckley

______________________________
Date
Student: Vishwa Kishore Mannem

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the project.

__________________________, Graduate Coordinator _________________________
Jinsong Ouyang                    Date

Department of Computer Science
Abstract

of

MODEL-BASED USER INTERFACE DESIGN

by

Vishwa Kishore Mannem

Model-based User Interface Design (MBUID) is a design approach where models play a key role in developing interactive systems. The main goal of a model-based approach is to acquire the knowledge and rules and to model the domain and generate a user interface. The process goes through several steps of refining before achieving a final and accepted user interface. In a MBUID process, focus is majorly on context-of-use where the interactive system is target to multiple devices.

This master project demonstrates how model-based approach can enhance the interactive applications in a multi-target environment. The project aim is to focus on supporting a design by addressing the change of platforms. The project also concentrates on the architecture of Cameleon Reference Framework (CRF), which is widely considered MBUID process that helps designers in creating user interfaces supporting multiple targets or multiple contexts. The case study is intended to give insight and deep understanding on the model-based interface design, benefits, challenges and future research in designing effective interactive applications. The project also aims at taking a use case system and implementing domain and task model, an abstract user interface, a
concrete user interface and a final user interface by visually representing the system design in multiple contexts of use.

_______________________, Committee Chair
Ahmed Salem

_______________________
Date
ACKNOWLEDGEMENTS

I would like to thank my project advisor, Ahmed Salem for his guidance and constant support throughout the project. His suggestions have been of great value for the project.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Acknowledgements</th>
<th>vi</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Tables</td>
<td>ix</td>
</tr>
<tr>
<td>List of Figures</td>
<td>x</td>
</tr>
</tbody>
</table>

## Chapter

1. **INTRODUCTION** ........................................................................................................1
   1.1 Purpose ..................................................................................................................2
   1.2 Personal Computers vs Smartphones .....................................................................3
   1.3 Problem Statement ..............................................................................................5

2. **MODELS** ....................................................................................................................6
   2.1 Tasks and Multi-targeting ..................................................................................8

3. **PROPOSED METHOD** ..................................................................................................11
   3.1 Model-based Design ............................................................................................12

4. **CAMELEON REFERENCE FRAMEWORK** .........................................................................14

5. **MODELING WITH CTTE** ............................................................................................17

6. **IMPLEMENTING UI USING MODEL-DRIVEN APPROACH** .........................................21
   6.1 Task Model ...........................................................................................................23
   6.2 Domain Model .......................................................................................................28
   6.3 Abstract User Interface .......................................................................................29
   6.4 Concrete User Interface ......................................................................................30
   6.5 Final User Interface (Smartphones) ....................................................................31
   6.6 Final User Interface (PC) ...................................................................................35
7. CHALLENGES AND OPPORTUNITIES ................................................................. 40
8. CONCLUSION ....................................................................................................... 41
   8.1 Summary ......................................................................................................... 41
   8.2 Future Work .................................................................................................... 42
References ............................................................................................................... 43
### LIST OF TABLES

<table>
<thead>
<tr>
<th>Tables</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Smartphone vs Tablet vs PC Growth</td>
<td>3</td>
</tr>
<tr>
<td>2. Editors used for Modeling</td>
<td>12</td>
</tr>
<tr>
<td>3. Sketching and Prototyping</td>
<td>13</td>
</tr>
<tr>
<td>4. CTTE Operators</td>
<td>19</td>
</tr>
<tr>
<td>Figures</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
</tr>
<tr>
<td>1. Mobile vs PC</td>
<td>4</td>
</tr>
<tr>
<td>2. Context of Use</td>
<td>7</td>
</tr>
<tr>
<td>3. Multi-Target Calendar</td>
<td>9</td>
</tr>
<tr>
<td>4. Multi-Target Fields</td>
<td>10</td>
</tr>
<tr>
<td>5. Cameleon Reference Framework</td>
<td>15</td>
</tr>
<tr>
<td>6. CRF for Multiple Contexts</td>
<td>16</td>
</tr>
<tr>
<td>7. Abstraction Task-Query</td>
<td>18</td>
</tr>
<tr>
<td>8. Abstraction Task-Login</td>
<td>18</td>
</tr>
<tr>
<td>9. Abstraction Task-Book a Car</td>
<td>23</td>
</tr>
<tr>
<td>10. Abstraction Task-Search</td>
<td>24</td>
</tr>
<tr>
<td>11. Interaction Task-Select_Options</td>
<td>24</td>
</tr>
<tr>
<td>12. Interaction Task-Pay_for_Car</td>
<td>25</td>
</tr>
<tr>
<td>13. Task Model - Car Rental System</td>
<td>26</td>
</tr>
<tr>
<td>14. Domain Model - Car Rental System</td>
<td>28</td>
</tr>
<tr>
<td>15. Concrete User Interface - Car Rental System</td>
<td>30</td>
</tr>
<tr>
<td>16. Final User Interface (1) – Smartphone</td>
<td>32</td>
</tr>
<tr>
<td>17. Final User Interface (2) – Smartphone</td>
<td>33</td>
</tr>
<tr>
<td>18. Final User Interface (3) – Smartphone</td>
<td>34</td>
</tr>
<tr>
<td>19. Final User Interface (1) – PC</td>
<td>35</td>
</tr>
<tr>
<td>20. Final User Interface (2) – PC</td>
<td>36</td>
</tr>
<tr>
<td>21. Final User Interface (3) – PC</td>
<td>37</td>
</tr>
</tbody>
</table>
22. Final User Interface (4) – PC ................................................................. 38
23. Final User Interface (5) – PC ................................................................. 39
1. INTRODUCTION

Considering the wide variety of devices currently available, designers face numerous challenges in developing intuitive interfaces. The number of devices tends to increase on a day-to-day basis. Due to the most recent advancements in technology industry devices such as smartphones, tablets, smartwatches, smart TV’s and virtual reality (VR) devices are coming into limelight because of the adoption rate and curiosity among users for these new devices. These devices are likely to increase in the coming years and building an interactive application for these systems is really challenging. There needs to be a model or a framework that can provide support in addressing the design challenges. The project aim is to focus on supporting a design by addressing the change of platforms. Irrespective of the device, the design should be in a consistent fashion across the devices. Stakeholders face numerous challenges when dealing with context-of-use. While designing an interactive application it is important to consider the following factors:

- Cross-device compatibility
- Consistency
- Usability and
- User experience

The cross-device compatibility is always considered a huge issue in a user interface development. The frameworks that are available on one platform may not be available on another or could be very different. Generating a consistent interaction among devices is also a challenging task. Due to the restrictions in hardware and software, tasks that are compatible in one device may or may not be compatible with other device. These limitations will challenge a designer in building an effective user interface by considering the above-mentioned factors.
1.1 Purpose

User Interface design plays a key role in the software development lifecycle process. Considering the amount of time and resources spent on user interfaces, it is a known fact that UI design is a time consuming activity in the development life cycle. The complexity and types of systems are increasing daily and are in fact becoming a challenging task for developers to develop the interfaces for multiple sources. The main purpose of an interactive system is to embrace a multi-target/multi-context-of-use environment.

As discussed, the focus is on three dimensions: platform, user and environment. For this purpose, throughout my project I will develop a car rental system. There needs to be a framework to address the challenges related to platform, user and environment. Following a model-based interface development will help the developers build interactive applications for a multi-context use. The usage of smartphones and tablets have emerged in the past decade or so. Due to the advancements in technology, portability is considered as one of the major advantage. Designing an interactive system for these portable devices has become a challenging task for designers where consistency is considered as a major issue. Recently many hybrid devices have come to the market that are more like a convergence of portability and productivity without compromising on usability experience. For example, Microsoft Surface Pro. The introduction of the iPhone in 2007 created a completely new device category among the devices that were already available and this is considered more of a convergence of portability and productivity. User experience of applications has been improving and becoming much easier for users to interact with a smart device on a day-to-day basis. Similarly, the introduction of iPad has also continued the trend with an introduction of new lineup among the devices.

Because of these new device trends and frameworks, we therefore are facing usability and other interaction related issues. Most of the websites are still have many design
challenges that are not compatible with the new device types and platforms. There is much more to be done in the user interface field in order to make these applications more usable and aesthetically pleasing to end-users on a cross-platform device.

1.2 Personal Computers vs Smartphones

Due to the introduction of new device types, the usage trend of PCs and laptops has been declining. According to Gartner’s recent report [1], the worldwide PC shipments declined by 8.3 percent in fourth quarter of 2015 which is considered as the fifth consecutive quarter of worldwide PC shipments decline. As shown in Table 1, for a single quarter smartphone shipments have reached a second highest level as the worldwide shipments reach 355.2 million in the third quarter of 2015.

Table 1. Smartphone vs Tablet vs PC Growth [1]

| MP3, Computer Ownership Has Dropped Among Younger Adults Since 2010 |
|-------------------------|---------|---------|---------|---------|---------|---------|
| % of U.S. adults ages 18-29 who own the following devices |
| 2010 | 2011 | 2012 | 2013 | 2014 | 2015 |
| Cellphone | 96 | 95 | 93 | 97 | 98 | 98 |
| Computer | 88 | 88 | 89 | - | - | 78 |
| MP3 player | 75 | 71 | - | 62 | - | 51 |
| Game console | 62 | - | - | 71 | - | 56 |
| Smartphone | - | 52 | 65 | 79 | 85 | 86 |
| Tablet computer | 5 | 13 | 32 | 36 | 40 | 50 |
| E-book reader | 5 | 8 | 27 | 24 | 23 | 18 |

Source: Pew Research Center surveys conducted 2000-15. Dashes represent years when these questions were not asked.

PEW RESEARCH CENTER
There has been many changes in the technology industry starting from mainframes to desktop PCs to mobile devices. Portability and mobility has been an important factor among the evolving devices and are encouraging many new hybrid devices types. As shown in Figure 1, combining IOS and Android smartphones, the device shipments are outselling PCs at 5:1.

![Mobile is the new scale](image)

**Figure 1. Mobile vs PC [2]**

Following are the trends [2] that Gartner is predicting in the near future:

- Traditional desktop computers sales will see a fall from 246 million in 2015 to 219 million in 2018
- Premium and basic ultra-mobile devices sales will grow from 241 million in 2015 to 290 million in 2018
- Mobile phone sales will grow from 1.9 billion in 2015 to over two billion in 2018.
1.3 Problem Statement

For an interaction device, the following challenges exist for a designer in building an effective user interface:

1. End-users: The interactive system designed may target a wide variety of users with different perspectives, preferences, culture and level of experience.

2. Devices: As there is a huge diversity among devices that are available on the market it is important to consider the wide range of device input capabilities. Achieving consistency among different versions to have a smooth transition across devices is quite a challenging task.

3. Programming languages: A designer has the flexibility to design a UI using different programming languages with different libraries.

4. Ability to respond: A designer has to consider the changes in the environment such as network connectivity, user’s location and other physical conditions.

5. The targeted system may have to serve a broad range of user categories, variety of devices such as tablets, smartphones, smartwatches etc. and target environment. It is a challenging task for developers to satisfy these design constraints.
2. MODELS

Models play a major role in capturing the knowledge about the world. Models help us to capture the meaningful aspects of the design by ignoring the tasks or activities that are not required. The design flow will become much easier for developers by following a methodological approach in designing a system. A well-designed model will help in designing effective interfaces that is independent of platform or device. Model-Based User Interface Development (MBUID) is one approach aiming towards the above-mentioned challenges. The Model-Based design helps reduce the developer effort in designing User Interfaces and increases the quality of a User Interface. MBUID approaches [3] also called as “model-driven” approaches where models play a major role in the development process.

The main goal of a model-based approach is to:

- Model the domain
- Create a task model
- Generate abstract model
- Generate concrete model depending on the context.
- Focus on the context and targeted devices.

A well-designed user interface should serve users’ needs by providing a consistent and common recognizable transition across devices. The design across multiple targets or multiple contexts of use should be easily understandable interactive elements without involving so many complexities. Context is composed of “con” (with) and “text”, refers to the meaning that from the adjacent text.
In a multi-target environment, the context usually differs by considering the following three combinations [4].

1. The users who are using the system
2. The hardware or software platform being used in the design process
3. The physical environment where the interaction actually takes place

Figure 2 depicts how the multi-target/multi-context of use is classified as:

![Diagram]

**Figure 2. Context of Use**

In a multi-target user interface, computation is done prior or on the fly. In a pre-computed user interface, everything is done in the design and implementation phase and then the interface adapts to a specific target based on the pre-computed instructions.
2.1 Tasks and Multi-targeting

The way tasks are performed on multiple devices could vary depending on the device restrictions and compatibility. A user may want to get the most critical tasks done irrespective of the device. However, performing these tasks can vary depending on the interaction elements that are used, presentation and the resources that are required. Considering the above fact, the choice of a device is important in order to perform a task. For example, a user wants to run software such as Photoshop, which is a resource demanding application and needs more configuration and RAM to perform the tasks smoothly. Considering this, it is more likely that the user opts for a traditional PC or a laptop. Whereas for tasks such as browsing the internet or checking email which often requires less resources, a small hand-held device such as a smartphone is often sufficient.

The relationship between interactions and tasks on different devices are as follows:

1. Performing the task with same interaction elements.
2. Performing the same task with different interaction elements. For example, in a car rental system, a pickup date from a phone could be different when compared to a desktop system. In Figure 3, we use different interactors: a calendar date picker on the left for a workstation whereas on the right a date picker for a mobile phone.
3. Decomposing the task into multiple tasks, Due to several resolution restrictions the user interface constraints always come into the picture that are device specific. For example, reading reviews on a phone or any other portable device can be cumbersome and can be decomposed or modified when compared to the desktop version. Whereas a desktop version can have reviews that can perform the task satisfactorily as there will be no performance related issues on a desktop computer when compared to a phone or other portable device.
4. Different sequential restrictions among subtasks. For example in Figure 4, a task such as filling out reservation form on a traditional desktop PC using one single screen without having any restrictions on the filling order. When compared the same task on a phone, due to space restrictions a user has to follow a certain order in order to get the reservation done.
5. Dependency within the tasks. For example, the online car reservation system assumes the user completes the reservation on a desktop and once it is complete the user will be given a reservation code and the same code can be used on a hand-held device and shown to the party whenever needed. This connection between devices will help the users in transitioning without any limitations.

6. Considering the type of task that a user performs. For example, a low cardinality field or task using a radio-button makes much more sense than using any other interactive element.
3. PROPOSED METHOD

The proposed model-based approach (MBUID) addresses the challenges in a multi-target environment. The process comprises a number of steps that helps designers in designing effective user interface applications for a multi-target/multi-context-of-use device.

- Step one; implement the high-level task modelling for a multi-target or multi-context of use application. In this step, designers have to put together a list of tasks that users perform.

- Step two; create and enhance the task models. The tasks that are mandatory in one target may not be required in another platform and so the designer has to evaluate the tasks accordingly. The task models here are designed using ConcurTaskTrees notation.

- Step three; design abstract user interface (AUI) from system task models. Here the major goal is to come up with description of the user interface composed of set of tasks and the various interactive elements that are used.

- Step four; generate the Concrete User Interface (CUI). This phase is totally platform-dependent and has to consider target device properties. The concrete user interface, articulates the user interface in terms of concrete interaction objects. The interaction elements defined here are implementation-language independent.

- Step five; conversion from concrete user interface to final user interface (FUI) happens here. The final interface defined in this step is implementation-dependent (Java, .NET, etc.).
3.1 Model-based Design

Considering the benefits of model-based design several frameworks came into existence for a better user interface design. For example frameworks such as, Hera [5], Roam [6] and Gummy [7]. Most of the work proposed by these models attempts to address the shortcomings of the design. In addition, it has become difficult for the developers to follow advancements in the technological standards and other device specific standards, which in turn creates a significant change in the contexts of use.

Table 2 lists the UI editors that are currently available for modeling and designing UI applications. The major purpose of these tools is to create and edit models. However, the only drawback is that they do not support user interface modeling. Table 3 lists the tools used for prototyping of user interfaces and sketching. However, these do not support interface modeling either. The modeling tools are useful in capturing the conceptual models of an interactive application in general, but they do not support with models specifically. Sometimes they do support with various user interface models but the context of use does not exist.

**Table 2. Editors used for modeling**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgoUML</td>
<td>Created by Tigris.org and is cross-platform compatible. (<a href="http://argouml.tigris.org">http://argouml.tigris.org</a>)</td>
</tr>
<tr>
<td>Dia:</td>
<td>Created by Alexander Larsson/GNOME Office and is cross-platform compatible. (<a href="https://live.gnome.org/Dia">https://live.gnome.org/Dia</a>)</td>
</tr>
<tr>
<td>MS Visio</td>
<td>Created by Microsoft and is compatible with Windows. (<a href="http://visio.microsoft.com/">http://visio.microsoft.com/</a>)</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Visual Paradigm for UML</td>
<td>Created by Visual Paradigm Int’l Ltd and is cross-platform compatible.</td>
</tr>
<tr>
<td></td>
<td>(<a href="http://www.visual-paradigm.com">http://www.visual-paradigm.com</a>)</td>
</tr>
<tr>
<td>Rational Rose XDE</td>
<td>Created by IBM and supports Windows, Linux and Unix platforms.</td>
</tr>
<tr>
<td>Balsamiq</td>
<td>It is a commercial tool used in creating high-level mockups and communicating design ideas. It is Cross platform compatible.</td>
</tr>
<tr>
<td></td>
<td>(<a href="http://www.balsamiq.com">www.balsamiq.com</a>)</td>
</tr>
<tr>
<td>JustInMind Prototyper</td>
<td>Commercial tool used for defining prototypes for web and mobile applications</td>
</tr>
<tr>
<td></td>
<td>(<a href="http://www.justinmind.com">www.justinmind.com</a>)</td>
</tr>
<tr>
<td>MAQETTA</td>
<td>A visual authoring tool of HTML5 user interfaces in the browser. It is open source software.</td>
</tr>
<tr>
<td></td>
<td>(<a href="http://maqetta.org/">http://maqetta.org/</a>)</td>
</tr>
<tr>
<td>Sketch Flow</td>
<td>UI prototyping tool to create interactive prototypes</td>
</tr>
<tr>
<td></td>
<td>(<a href="https://www.microsoft.com/silverlight/sketchflow/">https://www.microsoft.com/silverlight/sketchflow/</a>)</td>
</tr>
</tbody>
</table>
4. CAMELEON REFERENCE FRAMEWORK

Different frameworks were developed using the MBUID process over the last couple of years. One of the most widely considered MBUID is the Cameleon Reference Framework (CRF) [8]. The CRF framework helps designers classifying UIs supporting multiple targets or multiple contexts of use and the design and run-time phases of a multi-target user interface. Following this approach will give the designers a deep insight on context-sensitive user interfaces. The CRF framework as shown in Figure 5 is divided into four levels:

- The task and concepts level lists the hierarchies of tasks executed while interacting with the user interface.
- The Abstract user interface articulates the user interface in abstract interaction objects [8]. In this level, we consider the logical configuration and avoid the low-level details.
- The concrete user interface articulates the interface in terms of concrete interaction objects [8]. Here the user interface is defined more concretely using concrete interaction objects.
- In the final step translation happens to final user interface from concrete user interface and articulates the interface in terms of implementation-dependent source code (Java, .NET, etc.). The representation of a final UI can be in any programming language.

The CRF framework represents the model in a four-step process starting from Tasks and Concepts through Final User Interface. We can change the context of use depending on the user, platform or environment. The CRF framework highlights the need for plasticity in user interfaces. This produces for a multi-target UI that conserves the usability across diverse target environments. Instead of training and forcing the users to adapt to new
technologies and target environments, the CRF framework does all the work by adapting the interface based on the target device.

Figure 5. Cameleon Reference Framework [8]

To understand the framework, let us consider an example of a task, A Car Rental Reservation. The following car rental system is an example of the use of this framework. A task can further divide into several other subtasks, such as pick-up location, pick-up date (time), drop-off date (time). At the abstract user interface level, we first need to identify the user interface elements required to perform these tasks. In the next level, which is concrete user interface we need to determine the specific interaction objects
supporting the interface. For example, in a desktop interface a graphical list object can do the selection. Similarly, a calendar object can perform the date functionality as an interaction element. Identifying the more effective interactive elements and implementing them is part of the concrete user interface level. Finally, the final user interface, which is based on the choices based in the above levels. The context of use changes depending on the user, platform or environment. Considering the example above, a user should have the flexibility to book a rental car from a PC or smartphone or a tablet platform. Therefore, depending on the platform, the target is on context of use accordingly. As shown in the Figure 6, the two contexts defined are: desktop and mobile.

Figure 6. CRF for Multiple Contexts [8]
5. MODELING WITH CTTE

The ConcurTaskTrees Environment (CTTE) [9] is used for analysis and editing of task models supporting interactive applications. This is an open source product and the executable code is currently publicly available for download. The main purpose of having CTT notations is to overcome the limitations of previous notations used in the design of interactive applications.

The following notation uses CTTE. In CTTE, tasks are hierarchically organized; For example, in Fig. 7 the task “Query” is accomplished using two subtasks, “Provide Search Keyword” and “Show Result”. The relationship between the tasks is defined using the operation []>>, which tells us that the information is being passed from one task to another.

In a CTTE, there are four types of tasks:

- Abstract tasks
- User tasks
- Application tasks
- Interaction tasks.

A task is nothing but a set of actions performed by a user specific to a context of use to reach a goal. These tasks can further be decomposed into various sub-tasks. The tasks classified are as follows:

- In Figure 7, “Query” is considered as an abstract task. Similarly, in a system where user authentication is required the Login task from the Figure 8 is also an abstract task.
- A user typically performs all the tasks here without involving the interaction with the system.
In an interactive task, a user performs all the tasks with the help of the system. In Figure 8, a user providing a search keyword, entering user name and password considered as interactive tasks.

At the application level, the system usually performs the entire task without user intervention. In the Figure 8 as shown above, Show Result and ValiteUser considered as application tasks.

So there can be multiple tasks and relationships between these tasks based on the requirements and system complexity. The relationships between tasks differ based on the type and order of tasks that a user performs. Similar to the operator \[\bar{\triangleright}\] discussed above, the operator \[\Rightarrow\] means that a task can be performed in any order. The operator \[\bar{\Rightarrow}\] is a
sequence operator where the task before has to complete before moving on to the next task.

The main features of ConcurTaskTrees are as follows:

- Intuitive hierarchical structure often used to decompose a task into smaller tasks by maintaining the relationships. This structure provides a very granular detail of all the tasks and subtasks which in turn used as reusable task structures.
- Graphical syntax

CTTE allows the designers to focus on activities that are more accurate and relevant to the application from both a user and a system perspective. The tasks that are not needed or not mandatory are not considered.

Table 4 provides a list of operators used in a CTTE:

<table>
<thead>
<tr>
<th>Task_1</th>
<th>Operator</th>
<th>Task_2</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>&gt;&gt;</td>
<td>T2</td>
<td>Enabling</td>
</tr>
<tr>
<td>T1</td>
<td>[ ]&gt;&gt;</td>
<td>T2</td>
<td>Enabling + information passing</td>
</tr>
<tr>
<td>T1</td>
<td></td>
<td>&gt;</td>
<td>T2</td>
</tr>
<tr>
<td>T1</td>
<td>[ ]</td>
<td>T2</td>
<td>Non-deterministic choice</td>
</tr>
<tr>
<td>T1</td>
<td>π</td>
<td>T2</td>
<td>Deterministic choice</td>
</tr>
<tr>
<td>T1</td>
<td>[ &gt;</td>
<td>T2</td>
<td>Disabling (e.g. Form submit)</td>
</tr>
<tr>
<td>T1</td>
<td></td>
<td>T2</td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>---</td>
<td>-----------------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>=</td>
<td></td>
<td>Independence (any order, but finished)</td>
</tr>
<tr>
<td>T1</td>
<td>*</td>
<td>Iteration</td>
<td></td>
</tr>
<tr>
<td>T1</td>
<td>{n}</td>
<td>Finite iteration</td>
<td></td>
</tr>
<tr>
<td>T1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1</td>
<td>[x]</td>
<td>Concurrency + information passing</td>
<td></td>
</tr>
<tr>
<td>[T]</td>
<td></td>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td></td>
<td>Recursion</td>
<td></td>
</tr>
</tbody>
</table>
6. IMPLEMENTING UI USING MODEL-DRIVEN APPROACH

By designing the task model, a designer will come to know the specifications that are required for the abstract user interface. The steps involved in the implementation are as follows:

- **Step one**: Implement the high-level task modelling [10] for a multi-target or multi-context of use application. In this phase, designers have to come up with the list of tasks performed by a user. In this phase, designers have to come up with the list of tasks performed by a user.

- **Step two**: Create and refine task models that are required to interact for different platforms. The tasks that are mandatory in one target platform may not be required in another platform and so the designer has to choose and evaluate the tasks accordingly and identify the relationships among them.

- **Step three**: Develop the Abstract User Interface (AUI) from system task models. The major goal is to come up with description of the user interface composed of set of tasks and the various interactive elements that are used.

- **Step four**: Generate the Concrete User Interface (CUI). This phase is totally platform-dependent and has to consider target device properties. The concrete user interface, expresses the interface in terms of concrete interaction objects that are implementation-language independent.

- **Step five**: Translate CUI and express the interface in terms of implementation-dependent source code (Java, .NET, etc.). The representation of a final user interface can be in any programming language.

In a model-driven approach, user interface development undergoes several levels of fine-tuning, starting with no-contextual compatible to fully compatible with multiple contexts. The Cameleon Reference Framework (CRF) is the framework used to structure the user
interface design in four levels: tasks and domains, abstract user interface, concrete user interface and final user interface.
6.1 Task Model

This section provides a more complicated view of the car rental system. As shown in Figure 9, booking a car here is broken down into multiple sequential tasks. The tasks go in an orderly manner from Search a car to Show Confirmation. The information here is passed from one task to another by using the \[\to\] operator. Similarly, there could be a sequence of tasks where a user needs to complete before finishing the reservation process. The operator “\[>\]” indicates the interruption of a task. An example to interruption task is logging out of the system.

![Figure 9. Abstraction Task – Book a Car](image)

The abstraction task ‘Search a Car’ as shown in Figure 10 is further divided into multiple interaction tasks and application tasks as you can see below. The \[\parallel\] operator below used here to tell us that these tasks could happen concurrently.
Similarly, the Select_Options as shown in Figure 11 depict the interaction task, broken down into the options that a user can select from an interface. Figure 11 lists all the extra options that users can select while booking a car.

![Figure 10. Abstraction Task – Search](image)

The next abstraction task is the Pay_for_Car task. The task here has several interaction tasks where a user will enter personal and credit card details in order to finish the payment process. The Input_PaymentInfo interaction task is divided into three subtasks:

- **Input_PersonalInfo**, users are allowed to enter personal information such as Name, Phone, Email
- **Input_LocationInfo**, users are allowed to input details related to location such as billing address, country, zip.
- Input_Payment, users here input the details related to credit card such as credit card number, expiry, cvv and type. The TypeofCard interaction task is further broken down into two subtasks. The operator ‘[]’ used here denotes a choice task where a user can select one from the options available.

Once the above details are filled in, all the information that was entered in the previous tasks are then submitted using the Submit_Payment interaction task. The next final step is to submit the payment and show the confirmation to users. This is done using the Show_Confirmation application task. The Figure 12 below lists all the interaction and application tasks that are used in order to finish the payment and confirmation tasks.

![Figure 12. Interaction Task – Pay_for_Car](image-url)
Figure 13. Task Model - Car Rental System
Following is the XML generated code for the car reservation task model generated using CTT framework.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
<TaskModel NameTaskModelID="Book a Car"
 xmlns:coop="http://giove.isti.cnr.it/cttcoop"
 xmlns="http://giove.isti.cnr.it/ctt">
  <Task Frequency="" PartOfCooperation="false" Optional="false"
 Iterative="false" Category="abstraction" Identifier="Book a Car">
    <Name>name</Name>
    <SubTask>
      <Task Frequency="" PartOfCooperation="false" Optional="false"
 Iterative="false" Category="abstraction" Identifier="Search a Car">
        <Name>name</Name>
        <TemporalOperator name="SequentialEnablingInfo"/>
        <Parent name="Book a Car"/>
        <SiblingRight name="Select Options"/>
      </Task>
      <Task Frequency="" PartOfCooperation="false" Optional="false"
 Iterative="false" Category="interaction" Identifier="Select Options">
        <Name>name</Name>
        <TemporalOperator name="SequentialEnablingInfo"/>
        <Parent name="Book a Car"/>
        <SiblingLeft name="Search a Car"/>
        <SiblingRight name="Pay for Car"/>
      </Task>
      <Task Frequency="" PartOfCooperation="false" Optional="false"
 Iterative="false" Category="abstraction" Identifier="Pay for Car">
        <Name>name</Name>
        <TemporalOperator name="SequentialEnablingInfo"/>
        <Parent name="Book a Car"/>
        <SiblingLeft name="Select Options"/>
        <SiblingRight name="Show Confirmation"/>
      </Task>
      <Task Frequency="" PartOfCooperation="false" Optional="false"
 Iterative="false" Category="application" Identifier="Show Confirmation">
        <Name>name</Name>
        <Parent name="Book a Car"/>
        <SiblingLeft name="Pay for Car"/>
      </Task>
    </SubTask>
  </Task>
</TaskModel>
```
6.2 Domain Model

The Figure 14 shows the car rental system, which is illustrated using a class diagram. This domain model lists all the tasks that users perform irrespective of the platform that a user is. The model is the starting point in designing the AUI and CUI interfaces. The domain model and task model serve as a basis for the future developments in generating Abstract UI, Concrete UI and Final UI.

Figure 14. Domain Model - Car Rental System
6.3 Abstract User Interface (AUI)

An AUI as previously defined does not depend on any computing platform. It is independent of any interaction modality that includes vocal, graphical user interface, etc. At this level, we express how a UI needs to be designed in terms of interaction. The AUI is nothing but an interaction space that includes all the interaction elements irrespective of the computing platform in order to perform certain tasks. An abstract user interface is generated from models such as domain model, task model or a combination of both. UsiXML [11] defines the abstract user interface as a hierarchy of Abstraction Interaction Units, where each unit is responsible to perform a task or set of sub-tasks. An interaction unit is nothing but a group that executes all the logically connected tasks.

By definition, the abstract user interface design is independent of interaction modality. An abstract user interface can be generated from one or the combination of:

- The domain model
- The task model

An AUI is not required if:

- The requirements and target environments are known prior to the design of AUI, and
- The context of a user interface is already decided.

Different Concrete User Interfaces (CUI) can be designed from a single Abstract User Interface.
6.4 Concrete User Interface (CUI) - Smartphones

In this step, we define the concrete user interface for a multi-target environment. It expresses the interface in terms of concrete interaction objects that are implementation independent for a modality. Here the interface is defined in a more concrete way by defining the actual interaction elements. The Figure 15 defines the concrete user interface with actual interactive elements.

![Concrete User Interface - Car Rental System](image)

**Figure 15.** Concrete User Interface – Car Rental System
6.5 Final User Interface (FUI) - Smartphones

In this level, we define the final user interfaces for a multi-target environment. The following wireframes designed using Visual Paradigm for UML, will help navigate the application from one screen to another by performing different tasks on each page. The Figure 16, is a home screen which prompts users to enter the basic details such as pick up date, drop-off date, location and discount codes if any applicable. After entering the information, the user has to click ‘Next’ button in order to navigate to the next screen and perform the next set of tasks. The next screen, ‘Choose a Vehicle Class’ allows the users to choose a car that appears on the screen. Users are then prompted with the ‘Extras’ screen and then users input the personal information and credit card information on the next screens. On the final page users are prompted with ‘Submit’ button to finalize and submit the information in order to reserve a car. The series of tasks performed on a smartphone interface are listed as below:
Figure 16. Final User Interface (1) - Smartphone
Figure 17. Final User Interface (2) - Smartphone
Figure 18. Final User Interface (3) - Smartphone
6.6 **Final User Interface (FUI) – PC**

The final user interface for a desktop PC is shown in the Figures 19 through 23.

![Figure 19. Final User Interface (1) - PC](image)

Figure 19. Final User Interface (1) - PC
Figure 20. Final User Interface (2) - PC
Figure 21. Final User Interface (3) - PC
Figure 22. Final User Interface (4) - PC
Figure 23. Final User Interface (5) - PC
7. Challenges and Opportunities

Model-based technology is well suited for new interface requirements and will help the users by not bombarding them with multiple commands and options across device categories. It will help them to perform their tasks effectively irrespective of the device. Usage of small portable devices and smartphones to access the information is becoming part of the day-to-day life and there is in need to customize or scale the interfaces per the device. The main goal of a model-based design is to capture common semantically meaningful aspects of design.

There are many challenges involved with model-based design. One of the major difficulties that users face is to determine which interactive element or screen performs which task. As the complexity of an application increases so does the number of tasks making it difficult for a normal user to understand and remember the sequence of commands needed to perform a task. Microsoft Office is one example. Multi-platform support is one of the important concerns for traditional applications. Interfaces designed for traditional PCs are difficult to use in small screen displays. This eventually leads the developers to redesign for the new devices and by becoming an expensive transition from one target to another.
8. CONCLUSION

8.1 Summary

A typical interactive application may have many variations depending on the context-of-use. Therefore, an end user may run the application on any target device or the computing platform (on which the user is working). Similarly, the implementing language could be up to a user choice. It is also important to consider the factors such as, the organization a user belongs to, the user profile, the interaction modalities such as graphical interface or voice-enabled, etc.

It is a challenging task to develop user interfaces simultaneously for multiple contexts-of-use. Models help the designing process by implementing only the meaningful aspects of the design. Using the cameleon reference framework (CRF), designing an interactive application is done in four development steps: Tasks and Concepts, Abstract User Interface, Concrete User Interface and Final User Interface. Using the cameleon reference framework developers achieve an efficient interactive system by going through all the refining levels. The systematic designing done at various levels will help the knowledge transfer from one level to another and will open more room for designers in designing a product that is compatible across multiple devices. The task models defined using CTTE or other modeling environment will help the designers in concentrate the meaningful aspects of the design rather than the tasks that are not mandatory or optional to a specific set of targeted audience.
8.2 Future Work

User interface description languages (UIDLs) such as AUIML, UIML, XAML, XIML, XUL etc., have been introduced addressing the following user interface characteristics:

- Portability
- Device independence
- Multiple computing platforms

There were still issues related to these and with the integration of various models in the traditional graphical user interfaces [12]. It is a challenging task for designers to integrate the dimensions such as, models defined, target language and the software tools that exist on various platforms. We need a mechanism to express the design knowledge that would help the designer in the transformations from one context to another. To address the above-mentioned challenges, designers have determined a new language, USIXML. It is a XML-compliant User interface Description Language publicly available and lot of research has been going on for the past few years. With USIXML, language interface elements can be expressed and manipulated at every step of development depending on the context of use.
REFERENCES


