AUGMENTED FRAMEWORK FOR GENERATING DOMAIN-SPECIFIC MOBILE APPLICATIONS

by

RAMA KRISHNA RAJU RUDRARAJU

LEON JOLOLIAN, COMMITTEE CHAIR
HAIDER MOHAMMAD
MURAT M. TANIK

A THESIS

Submitted to the graduate faculty of The University of Alabama at Birmingham,
in partial fulfillment of the requirements for the degree of
Master of Science in Electrical and Computer Engineering

BIRMINGHAM, ALABAMA

2017

AUGMENTED FRAMEWORK FOR GENERATING DOMAIN-SPECIFIC MOBILE APPLICATIONS

RAMA KRISHNA RAJU RUDRARAJU

ELECTRICAL AND COMPUTER ENGINEERING

ABSTRACT

In this research, we developed a new approach called Augmented Framework to mean the integration of existing frameworks with domain-specific knowledge. The resulting framework provides a productive development environment suitable for app-developers to generate customized mobile-applications in the domain chosen by the framework developer. This process requires the application developer to provide specifications about the domain for customizing the generated mobile-applications. With this approach, the entire process of developing a mobile-application becomes simplified since it will only need minimum design and development effort.

The approach for developing an Augmented Framework consists of three stages. The first stage consists of the development of a variety of user-interfaces related to the chosen domain. In the second stage, the approach illustrates a mechanism for dynamically integrating the customized components into the framework. The third stage involves the development of the logic to display the appropriate user-interfaces based on the customized components. The three-stage approach results in generating an Augmented Framework.

There are three distinct benefits for using the Augmented Frameworks: a) a significant reduction in application development time, b) an overall decrease in the cost of creating the application and c) an ability to customize features of the mobile app. Furthermore, Augmented Frameworks developed by this approach will follow generally

accepted standards and common-sense options for their intended domains application. The framework-developer may impose some restrictions in the way applications are generated. Therefore, the application-developer will be asked to follow prescribed design specifications associated with the framework. In return, it will result in a fast and precise application development process.

The effectiveness of the proposed approach is illustrated with a case study, which details the steps involved in the creation of an Augmented Framework. The chosen domain for this case-study is resource allocation for reservation systems.

Keywords: Augmented Framework, Application developer, Domain-Specific, and Framework developer.

DEDICATION

I dedicate this thesis to my parents and Dr. Leon who has provided support throughout my graduate journey and always had faith in my success.

ACKNOWLEDGMENTS

I thank my parents Ramesh Babu & Sujatha and my brother Seetha Ram for investing their lifetime savings and for their moral support. I would like to thank my relatives and mentors Ramu Kosuri, Jagadesh Sagi, Rama Raju Rudraraju, Prasad Pinnamaraju, Srikanth Karra, Viswanadha Raju Datla, Annapurna Rudraraju, Anil & Ravi Rudraraju, Kishore Kanumuri, Sreekanth Paricharla and many more for their enormous help and mentorship.

I would like to thank my advisor and committee chair Dr. Leon for his countless hours of guidance, reading, encouraging, and most of all patience throughout the entire process. His guidance helped me in all the time of research and writing of this thesis. Despite of his busy schedule as an associate chair of the department, the door to his office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this research to be my own work, but steered me in the right direction whenever he thought I needed it. I could not have imagined having a better advisor and mentor for my Master's study.

Finally, I would like to thank all the people that have supported me on this journey: Dr. Murat Thanik, Dr. Karthikeyan Lingasubramaniyan, Dr. Haider Mohammad, Sandra Mohamad, and many others.

If I forgot to mention any others, it is due to my negligence.

TABLE OF CONTENTS

Page

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

Page

LIST OF FIGURES

Figure *Page*

LIST OF ABBREVIATIONS

CPU	Central Processing Unit
CSV	Comma Separated Value
DSL	Digital Subscription Line
GUI	Graphical User Interface
HTML	Hyper Text Markup Language
IPO	Input-Process-Output
MP3	MPEG-2 Audio Layer III
MVC	Model View Control
RDBMS	Relational Database Management Systems
SQL	Standard Query Language
UI	User Interface
VRML	Virtual Reality Model Language
VR/AR	Virtual Reality / Augmented Reality
XML	Extensible Markup Language

CHAPTER 1

INTRODUCTION

This chapter introduces the problem that is the subject of this research and the overview of the proposed framework. Section 1.1 describes the problem associated with the development of complex applications, the motivation behind the proposed framework and the fields on which it will show its impact. Section 1.2 gives an overview of the approach we are following and introduces different stakeholders involves from framework development process to application development. Section 1.3 illustrates the importance of proposed approach and the benefits of using it. Section 1.4 gives an overview of some domain-specific frameworks.

1.1 Problem and Motivation

Programming was first used to develop small systems with limited amount of functionality. As developers started to realize the importance of programming, it was soon applied to many domains. As the programming continued to be applied to increasingly large problems, the level of difficulty in programming is also increased. In recognizing the growing complexity of software development, *Frank DeRemer* and *Hans Kron* described two different levels of programming in their 1975 book: "Programming-in-the-large versus Programming-in-the-small" [1]. They also discussed the complexity of writing large programs and the inherent difficulty in using the same approach. They defined the code

into a set of small modules to make it easy to code, easy to understand, and even easy to edit. In 1986, *Fredrick P. Brooks* in his book "No Silver Bullet - Essence and Accident in Software Engineering" [2] points out that large software system will face essential and accidental difficulties. Current systems play a remarkable role in eliminating the accidental difficulties like high-level languages, time-sharing, and workstations. Furthermore, in his 2007 panel paper [3], *Brooks* mentions that at least half of the remaining trouble is essential and had an inherent complexity.

Over the past few decades, many tools have been proposed to cope with the development of "large and complex" software systems. Some of the most successful tools developed for dealing with this complexity are frameworks [4], Object-oriented design [5], and Component-Based model [6].

More recently, a growing computing trend has emerged with the increasing use of smart devices, such as smartphones and tablets. The software running on these devices has the following characteristics:

1) Programming-in-the-Small.
2) Limited Hardware, CPU, and Memory.
3) Access to the cloud.

To cope with the development of applications for these devices, a verity of frameworks have been developed. However, the focus of the frameworks has been targeted towards components such as Graphical User Interface (GUI), Database, Cloud, etc.

1.2 Approach Overview

The primary aim of this research is to develop a new approach for creating frameworks that extend the capability of existing frameworks with domain knowledge. We refer to this new type of framework as an augmented framework. The approach consists of two phases as shown in Figure 1. The first Phase involves creating an Augmented Domain-specific framework. This consists of taking an application development framework and adding some domain-specific knowledge to it by the framework-developer. This results in the development of a framework with embedded domain knowledge. We call the resultant framework with integrated domain knowledge as a domain-specific augmented framework. The framework developed by this process allows external components provided by the application-developer (for customized behavior) to integrate with the framework at run-time to generate a customized application. This model keeps core Framework functionalities intact and even allows reuse in other environments or applications. The second phase involves having the app developer add customized components to the framework to generate the customized application. The integration of customization components with the framework is accomplished using a mechanism called Dynamic-Binding.

Developing an Augmented Framework for Generating Domain-Specific Applications requires a deep understanding of the specific domain, as well as a reference architecture that can be used as a basis for the design of the applications. The proposed development guidelines will outline the steps required to create a domain-specific augmented framework. The guidelines include steps on how to: develop the requirements, design the user-interface templates, usage of a mechanism to add the external data, and the

definitions of software components associated with the chosen domain need to include in the framework. As a case-study to demonstrate the effectiveness of Augmented Framework, we developed a framework for the case of resource allocation.

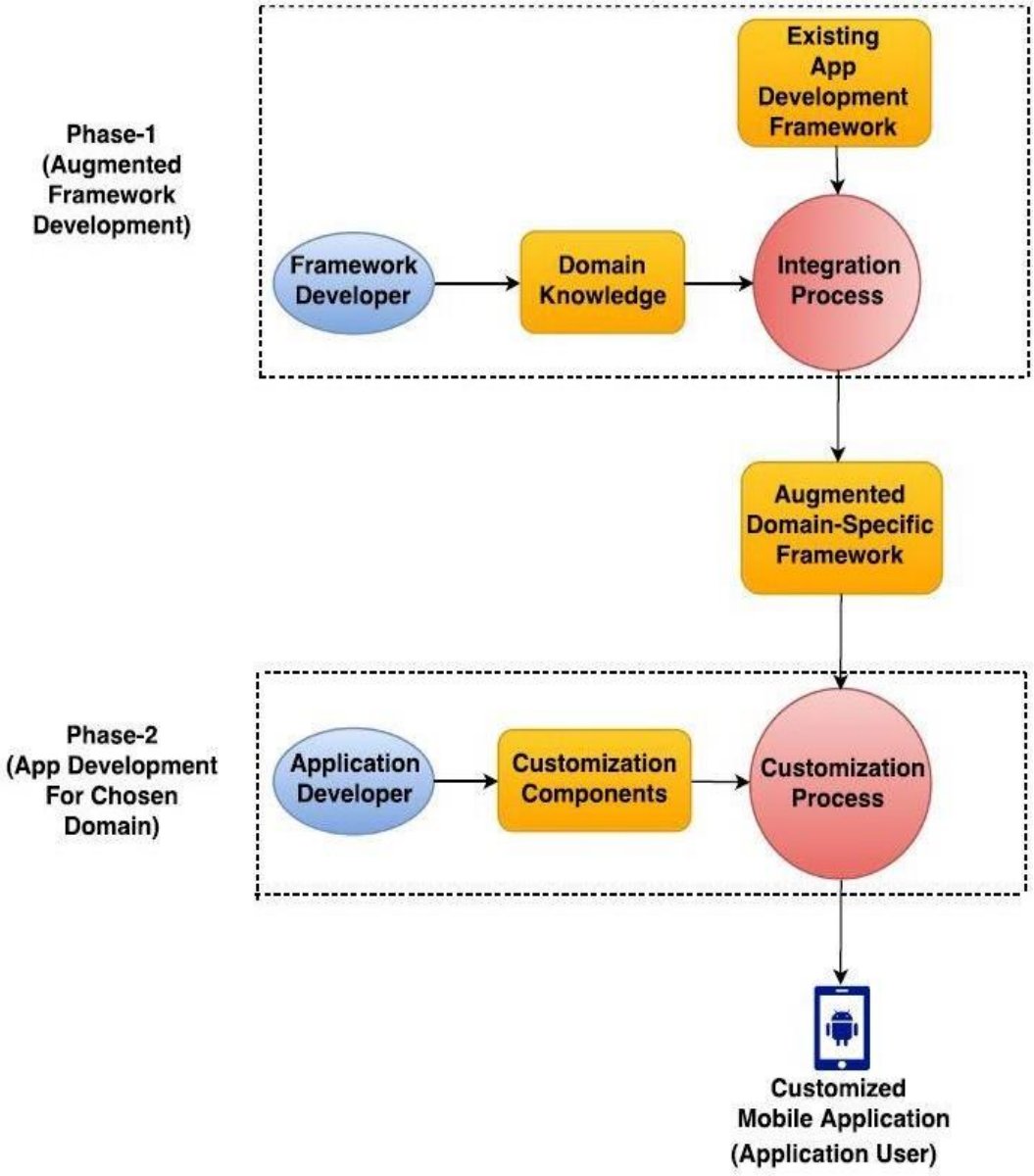


Figure 1: Workflow and Stakeholders

In the context of this Augmented Framework, we recognized three distinct roles among all stakeholders:

a) The Framework developer:

We refer to the framework-developer as the individual who follows the guidelines for creating the domain-specific augmented framework. The framework is capable of taking external data as input and later generating the customized application.

b) The Application developer:

We refer to application-developer as the person who uses the domain-specific augmented framework and adds some extra code to generate customized applications.

c) The Application User:

Finally, Application-user is the end user who uses the generated application to perform his tasks.

1.3 Importance of Domain-Specific Augmented Frameworks

Increasingly, there is a great need to develop complex applications, which can run on mobile devices. There are many traditional mobile application development frameworks. While these frameworks follow the "Programming-in-the-large" model, however, they are focused on simplifying the programming task as it relates to the devices. These frameworks do not add the domain knowledge. Traditional frameworks require the deep understanding of multiple fields like scripting languages for user-interface, programming languages to generate dynamic effects, event handling, and SQL as well as RDBMS concepts for database operations that make the application development process complex and time-consuming. Since in our Domain-Specific Augmented Frameworks we

are using “programming-in-the-small,” it makes easy to create the application. Also, we are building the framework based on a specific domain which makes it straightforward to generalize and perfect the functionalities that are most common in the chosen field. The apparent benefits of applying this proposed approach is a significant reduction in development time and an overall decrease in the cost of creating and coding the customized application when compared with developing similar applications by using regular frameworks from scratch.

1.4 Overview of Domain-Specific Frameworks

Domain-Specific frameworks have been traditionally developed to assist software development by taking advantage of reusable components. However, those frameworks are not suitable for mobile devices due to limitations in the processing power of mobile devices. This section presents an overview of three different domain-specific frameworks that are used in their respective domain.

The AIBench Framework:

In the field of scientific software development, one of the widely used framework is *AIBench* [7]. This framework adapts latest concepts like MVC, and there are well-defined coding section like operations, data types, and views. Since framework has distinct parts to insert the code, the software developer can write the corresponding algorithm and easily plugs it into the framework. This allows the programmer to keep clean his own application core functionality and even reuse it in other environments or applications. It results in a customized application with fixed four window user interfaces. AIBench model

is perfectly suitable for any scientific experiments that are having the following functionality like commonly contains collecting parameters, executing an algorithm, and analyzing the results. Although AIBench can be classified as a white-box framework (which require the programmer to inherit or implement specific classes or interfaces), the end application code (algorithms, data structures, and viewers) contain the minimum amount of framework-related code.

The Software Factory Framework:

Software factory [8] is mainly used in the domains where there is continues need to update the existing software. For example, in the industries like Financial, Banking, and Healthcare if there are a small portion of software update (Change of Numerical Data) results in necessary updates in many parts of the application code. Software factory reduces the code and time to generate updated applications. This framework requires the deep understanding of concepts like Component-based development, model-driven development, knowledge on the product line and Domain Specific Language.

The VHD++ Framework:

VHD++ [9] is a high-performance framework applied in the development of interactive VR/AR applications. VHD++ contains inbuilt packages to simplify the application development process. For the regular VR/AR applications, there is a repository available, which contains the commonly used code arraigned in a prescribed manner. To access the external components VHD++ has its own *vhdServices*, pluggable portion, etc. Once the external components are integrated into the framework, it has a *vhdRuntimeEngine* responsible for binding components with the framework architecture

and generates an interactive VR/AR application. This application process little bit but the application development process is fast and reliable.

1.5 Summary

In this chapter, we introduced the background associated with the software development process. We mentioned about the challenges faced in the development of the software if we apply traditional approaches (*Programming-in-the-Large*) to all kind devices without considering the limitations of the device. Some of the examples of that kind of devices are smartphones and tablets. To solve this problem, we introduced a new approach, which extends the capabilities of the existing frameworks we call it as Augmented Framework. We extended the capabilities of existing frameworks by integrating the domain knowledge into the framework. This chapter also mentions the importance of domain-specific Augmented Framework, and also an overview of other domain-specific frameworks.

CHAPTER 2

A SURVEY OF DIFFERENT DOMAIN-SPECIFIC FRAMEWORKS

The overall idea of this chapter is to give an inside into the other domain-specific frameworks. This chapter contains a detailed study of three widely used frameworks and why they are applicable in their respective domains. Section 2.1 explains about AIBench used in scientific software development. Section 2.2 gives an inside into the Software factory applied for product-line development. Section 2.3 contains a detailed study of VHD++ applied in the development of interactive VR/AR applications.

2.1 Framework for Scientific Software Development

2.1.1 Characteristics of Scientific Software Development:

In general, most of the scientific experiments have the same workflow even with the change of inputs and output. Since the researcher's focus is on the core experimental study, it is always better if any generalized framework can provide them with a quick and reliable user interface to increase the productivity of research groups. AIBench best fit because of the IPO (input-process-output) model by means of the MVC (model-view-controller) design pattern and makes use of several technologies (XML, Java 1.5, etc.). The framework is a plugin-based architecture with contains three types of inputs: operations, data-types, and views.

2.1.2 The AIBench Framework:

AIBench is intended for use in scientific experiments that are having the following functionality like commonly contains collecting parameters, executing an algorithm, and analyzing the results. The framework is a plugin-based architecture with contains three types of inputs: operations, data-types, and views as shown in Figure 2. The model operation part contains the core algorithm on which logic of a scientific experiment will reside. It is inserted in the form of Java Class with well-defined input and output ports. In this model, Data-Type supports data-structures types associated with core experimental program requirement. Sometimes data-types may be non-primitive or classes but should contain information, which is easily assessable by the process to use in its operations or to display in GUI. In this model, there is a default four-window structure, where each window has its properties. Based on the requirement, external programmer can add experiment specific UI components like input (from Console, TextViews, etc.) and output models (Tables, Graphs, etc.).

Although AIBench can be classified as a white-box framework (which require the programmer to inherit or implement specific classes or interfaces), the application specific code (algorithms, data structures, and viewers) contain the minimum amount of framework-related code. This allows the programmer to keep clean his application core functionality and even reuse it in other environments or applications (web, a more detailed GUI, etc.)

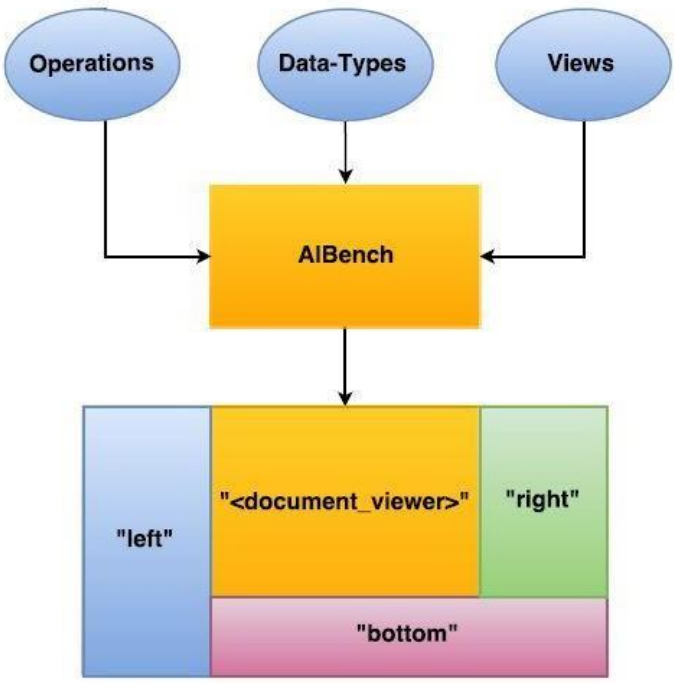


Figure 2: Input and Output formats in AIBench

2.2 Framework for Product Line Development

2.2.1 Characteristics of Product Line Development:

Usually, the development of updated versions of software is done over a considerable period. In some industries like financial services, banking, and healthcare slight changes in the software is necessary over short periods. For this kind of scenarios, if they follow normal software product-line(Next Version) development processes based on component-based or software product line tents, they involve iterating through each portion of the software. Therefore it results in a consumption of a lot of time as well as money for small software changes. In the enterprise software, development to cope up with this kind of scenario researchers in Microsoft proposed a framework called Software Factory.

2.2.2 The Software Factory Approach:

The software factory is one the most sophisticated and advanced frameworks. It has major applications in product-line development industries. It is the combination of component-based development, model-driven development, and software product line development. Its main aim is to reduce the cost and time of development Enterprise applications having product-lines (Need of continues updated applications).

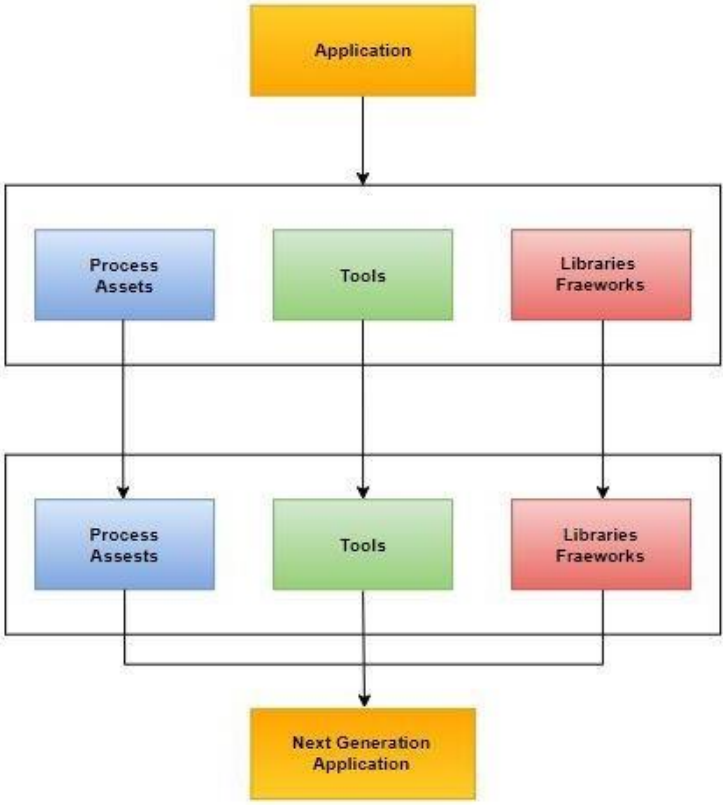


Figure 3: Process involved in the development of Product line development

The software factory follows knowledge bases methodology, where it systematically captures the knowledge of how to produce the member of a specific product family. It stores that information in the form of assets, tools, libraries frameworks, etc. as shown in Figure 3.

To generate an application the asserts, tools, libraries of the previous application is collected into an Interactive Development Environment (IDE), such as Microsoft's Visual Studio.NET. When IDE's are configured in this, way they will become software factories. In other words, developers build assets which are used as plugins to IDE'S which generates next family products. Since applications generated from software factory is more like providing updated versions to an existing application, it uses a technique called Domain-specific language (DSL) which reduces the complexity of the process.

We can say that Software factory is a model-driven product line defined by domain specific language. It plays a good role in the development applications in financial services, banking, and healthcare application development where there are necessity risk calculation and forecasting engines. The main drawback of this framework is it will demand a lot of multi-field expertise like Component-based development, model-driven development, knowledge on the product line and Domain Specific Language to generate an application. Also, it will not support multiple suppliers to collaborate in the development of the product.

2.3 Framework for Developing Applications in VR/AR domain

2.3.1 Characteristics of VR/AR Applications:

Nowadays, Virtual Reality and Augmented Reality applications is a rising trend in the motion picture industry. The application development process follows traditional development process of accessing the external repository manually and writing the code to fit the external components to generate the VR/AR applications. In present days to maintain the demand the application development process should have the following characteristics:

deliver new, always faster, and always more in short period. VHD++ plays a remarkable to cope up with those characteristics in the VR/AR domain. VHD++ has a well-built package containing most widely used components, and it provides services to access the traditional repositories.

2.3.2 The VHD++ Framework:

In current days, there is a rapid advancement in the graphics as well as virtual character simulation technology which helps in the development of interactive audio-visual simulation applications. VHD++ is an Object-Oriented Framework, which provides an interactive environment to different components like audio, video, etc.

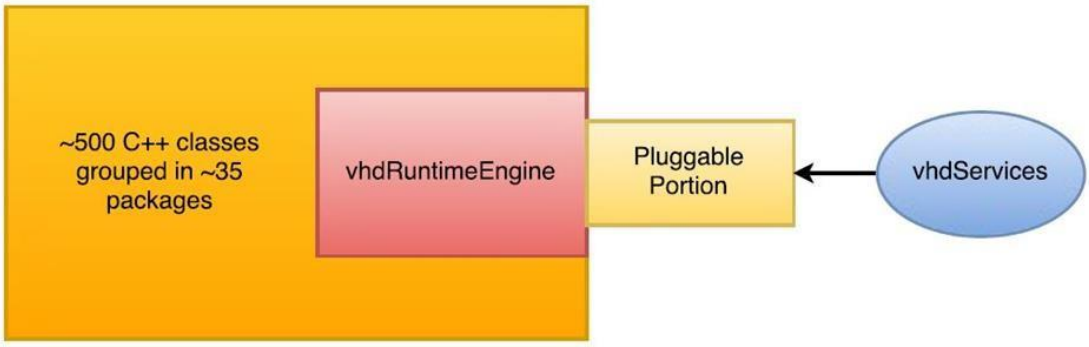


Figure 4: Mechanism involved in the VHD++ Framework

As shown in the above Figure, VHD++ primarily contains nearly 35 packages containing approximately 500 classes. It also featured some high-performance component called *vhdRuntimeEngine*. The *vhdRuntimeEngine* has a mechanism to inherit the external components from the VR/AR simulation domain. *VhdRuntimeEngine* has an interface called pluggable portion, which allows the external components that come from *vhdServices.* The external components in the VR/AR simulation domain are in the form of component-based structure. The components in the VR/AR repository contains a well-

defined naming convention [10] for storing the data. Since VR/AR is component based, the naming convention is based on the functionality (XML, VRML, WAV, MP3, etc.) [11] each of them is further divided into based on the area to which they are applied. VR/AR repository components are fit for reuse and tested thoroughly. *vhdServices* are the software components are responsible for carrying the VR/AR components into the VHD++ framework. One the *vhdServices* delivers the external components it is the *vhdRuntimeEngine* in processing the information and adding it to the necessary locations. *VhdRuntimeEngine* has a robust nature to communicate with other *VhdRuntimeEngine* in different systems at the same time.

VHD++ is a complex and domain-specific system; to make effective use of it, the application developer has to know so many inside out of the in VR/AR domain. Like any other component based system VR/AR repository grow when application developer exports the code that has the potential to reuse into the pool by following the naming convention.

2.4 Summary

In this chapter, we presented three widely used domain-specific frameworks *(AIBench, Software Factory, and VHD++)* that are useful for the development of applications in Scientific, Product line and VR/AR domains. Initially, we explained the characteristics of each of the three domains and why these frameworks are best fit for that domain. We presented in-depth details on the mechanism used in the framework and the process of generating applications from frameworks.

CHAPTER 3

APPROACH FOR DEVELOPING A DOMAIN-SPECIFIC AUGMENTED FRAMEWORK

In this chapter, we outline the guidelines for the development of a domain-specific augmented framework. Section 3.1 gives a complete overview of the approach we are following in the development of the augmented framework. Section 3.2 illustrates the development of all kinds of graphical user interfaces. Section 3.3 illustrates the approach we are following to integrate the customization components into the augmented framework. Section 3.4 specifies the guidelines for displaying the appropriate user interfaces based on the customization components that comes during runtime.

3.1 Overview of the Approach

The proposed approach describes a step-by-step process for the development of a domain-specific augmented framework. As shown in Figure 6, the proposed approach has three stages: In the first stage, we provide the guidelines for the framework-developer to create the design and skeleton code for generating the user interface; this allows customization of the screens by the application-developer. In the Second stage, we provide the guidelines to the framework-developer to create the dynamic binding mechanism to integrate the external customization components at runtime. The third stage consists of guidelines to display the appropriate user-interface based on input at runtime

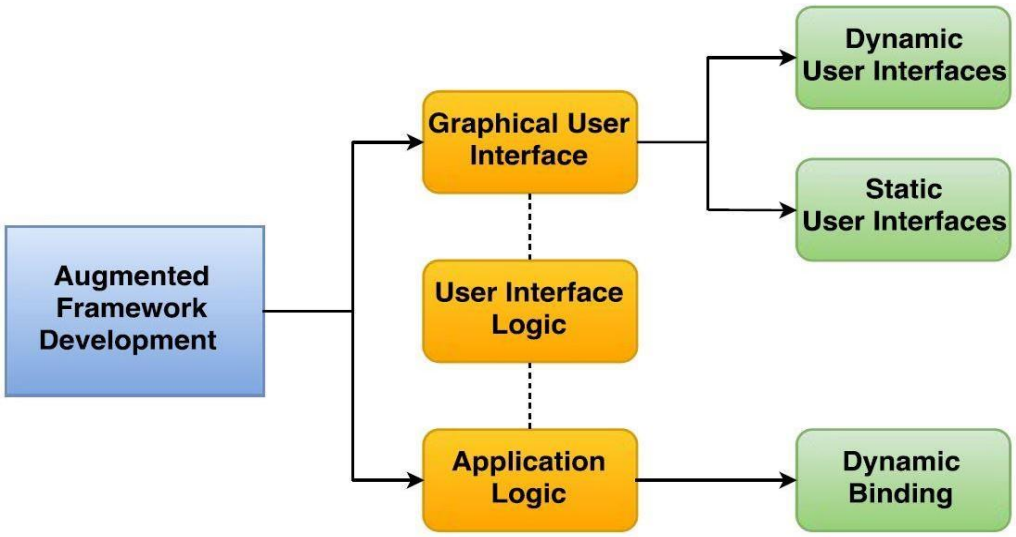


Figure 5: Stages in the Development of Augmented Framework

3.2 Graphical User Interface Development

Usually, user interface development (Web or Mobile), the creation of the skeleton means creating the user-interfaces without any value attached to the elements in the user-interface. Since in our framework we allow customization, the values of the components in the user-interface are populated with the values given by application-developer at runtime.

In our approach, the user-interface development is categorized into two types:

1) Static user-interface: This refers to the development of screens that can be included in the app without the need for customization. For example, the login, Signup, admin, etc. screens.

2) Dynamic user-interface: This refers to the development of screens that requires some input specifications from the app developer.

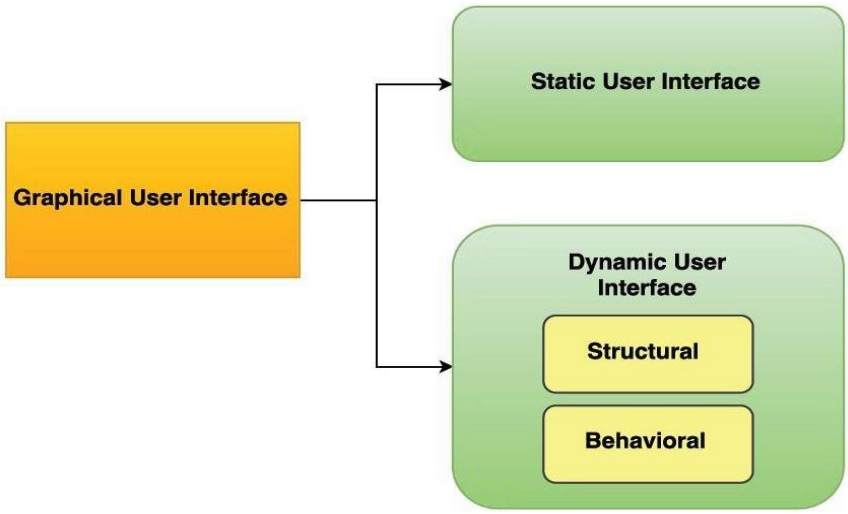


Figure 6: Two types of Graphical User Interfaces

3.2.1 Static User Interface Development

The static user interfaces are the user interfaces, which are not affected by the resource type or resources structure. For example, the user-interface associated with authentication, signup, admin functionality, will remain static across many applications.

As an example, when apps provide a signup or login screens, it almost invariably consists of two fields and a couple of buttons. The two fields are used by the user to provides their username and password. One button is used to submit the login information, and another button is used to navigate to the signup screen for the new user.

3.2.2 Dynamic User Interface Development

Dynamic user-interfaces are one of the essential components in the development of frameworks that supports customized applications. The dynamic user interfaces are the user interfaces, which varies based on the resource type or resources structure. Dynamic user-interfaces are categorized into two types of user-interfaces: behavior-dependent and structure-dependent. In the following paragraphs, we describe these two categories.

3.2.2.1 User-Interfaces with Structural Dependence

Since the framework can manage any number of resources, there is a need to create layouts, which changes according to the number of resources. For example, if the application-developer needs to manage three resources, the user-interface has to show three buttons with corresponding resource names. If the application-developer wants to manage one resource, it should show only one resource.

For example, in our case study to be discussed in the following chapter, the framework generates a screen with a button for each of the resource introduced by the app-developer. Each of the buttons, when pressed will trigger the corresponding screens that display the needed details about the resource.

3.2.2.2 User-Interfaces with Behavioral dependence

While developing the user-interfaces for this category, the framework-developer need to look at different kinds of scenarios associated with the chosen domain where the framework is applied. For each scenario, there are different kinds of inputs that need to be managed. Therefore, there is an obvious need of different user-interfaces.

For Instance, in our case study (Chapter 4), we demonstrate the behavioral-dependency user-interfaces with the screens associated with the time allocation of a resource. There are resources that are allocated to users on an hourly basis. The app has to generate each type of screen associated with the resource based on the input received by the developer.

3.3 Application Logic Development

In software development, the application logic is a critical and complex part. It defines the actions associated with the applications, such as handling the *onclick* operations, taking the input from the screens, creating the databases and performing the Database related tasks. In our augmented framework development, our main contribution is in the development of a mechanism to accept external data we call it "dynamic binding."

3.3.1 Dynamic Binding

In the augmented framework development dynamic binding means developing a mechanism which allows external data into the framework as input at runtime. It is a crucial component in the framework development because it the path through with customization data will be integrated into the framework.

The Augmented framework should be developed in such a way that it will take two kinds of inputs (Text Files) from the application developer at runtime. The first type of textfiles is the "Resource-Listfile" which contain the number of resources as well as the names of the resources files on which the data is present. The second kind of text files is

the “Data-and-Behavioural-File” contains the data and its behavior in the form of methods. In the following sections, both of the input formats are described.

3.3.1.1 Resource-Listfile

The Resource-Listfile contains the information regarding the *Structure* of resources and the names of each resource the customized application is going to manage. First, the framework-developer have to decide the pattern of Ressource-Listfile. Once the pattern is decided the framework developer has to write the logic to extract the list information and navigation to the corresponding files given by the application-developer. Framework-developer should also mention the location in the framework where the application-developer should add the resource file. More details regarding the Resource-ListFile integration is mentioned with the case study described in Chapter 4.

3.3.1.2 Data-and-Behavioral-File

The “Data-and-Behavioral-Files” are the files which contain data associated with the resource and its behavior. To follow a similar pattern in the input resource files, the framework-developer has to provide a fixed input standard so that every application-developer will follow it. Nowadays, we see so many frameworks which accept external data to develop customized applications. They follow the traditional approach of writing the code in the IDE provided by the framework.

In our approach, we are not restricting application-developers to follow any developing environment; they can simply write the code on a “Text Document” and add int the framework. Since we are not using any traditional IDE, the text document

(Containing the Code) should undergo the following six steps as shown in the figure. The code associated with each of the steps as shown in Figure 7.

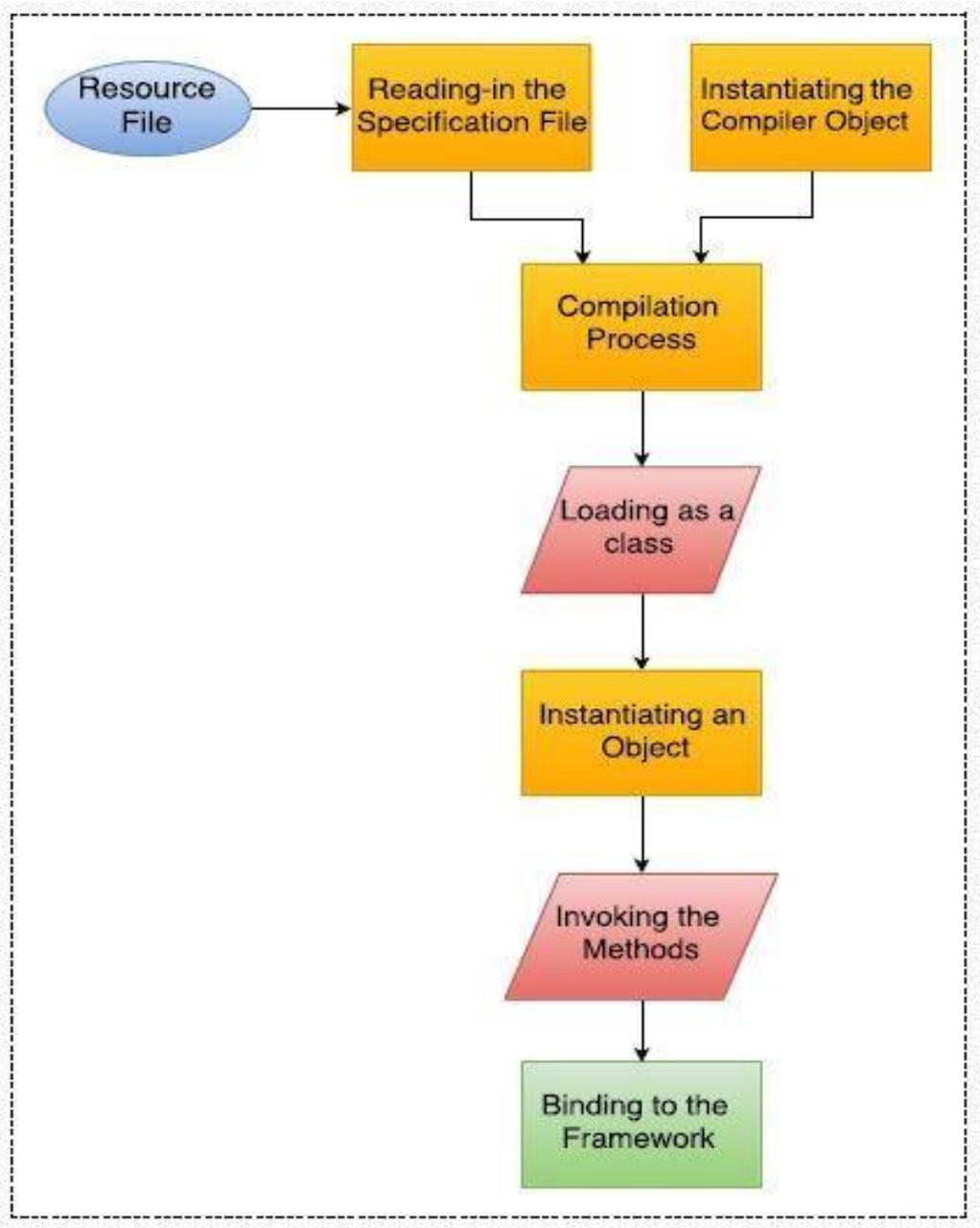


Figure 7: Steps involve in Binding the "Data-and-Behavioral-Files"

Step 1 → In this step involves reading the specification file which contains the customization components.

Step 2 → This Step involves instantiating the compiler object of the associated programming language to convert the source code from text files to a lower level language (e.g., assembly language, object code, or machine code) to create an executable program."

Step 3 → From Step 1 and 2 we got the necessary files as well as compiler, this step involves compiling both together to generate a bytecode

Step 4 → This step involves creating a class from the input file, to facilitate the necessary operations.

Step 5 → This step involves instantiating an object to the class that we had created in step 4 access the customization content in it.

Step 6 → This step deals with invoking both the data as well as behavioral methods in the input files.

The code associated with above six steps is illustrated with the following example:

Figure 8 illustrates a Text Document File with a variable and a method with file name as "Test.java"

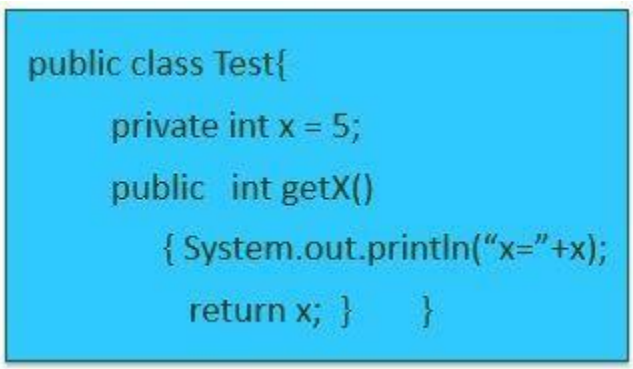


Figure 8: Code in the Text Document

Figure 9 illustrates all the Six steps involved right from Accessing the source code in the text document to Invoking the methods in the source code.

```
File sourceFile = new File("Test.java");                                  // Step 1

System.setProperty("java.home", "C:\\Program Files\\Java\\jdk1.8.0_151"); //
JavaCompiler compiler = ToolProvider.getSystemJavaCompiler();             //  } Step 2

compiler.run(null, null, null, sourceFile.getPath());                     // Step 3

Class params[] = {};
Object paramsObj[] = {};
Class thisClass = Class.forName("Test");                                  // Step 4

Object iClass = thisClass.newInstance();                                  // Step 5

Method thisMethod = thisClass.getDeclaredMethod("getX", params);          //
thisMethod.invoke(iClass, paramsObj);                                     //  } Step 6
```

Figure 9: Code associated with all the six steps

When we execute the above code as shown in Figure 6, we will get the following output:

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\rrkr2016>cd desktop

C:\Users\rrkr2016\Desktop>cd DynamicBinding

C:\Users\rrkr2016\Desktop\DynamicBinding>javac a.java
Note: a.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

C:\Users\rrkr2016\Desktop\DynamicBinding>java a
x=5

C:\Users\rrkr2016\Desktop\DynamicBinding>
```

Figure 10: Execution Process

Figure 10 shows the proof of accessing the content in the external file by following the proposed Six-step approach. The code in Figure 9, will take the input from the source file (Figure 8) and invoked the method in it.

3.4 User-Interface Logic

From Graphical User interface and Application logic section, the framework will get the user interfaces and the mechanism to extract the customization data from application-developer. This stage consists of writing logic to display the appropriate user interfaces based on the structural and behavioral data. As mentioned in section 3.3, the structural data comes from "Resource-Listfile," and behavioral data comes from "Data-and-Behavioral-Files." The *Resource-Listfile* and *Data-and-Behavioral-Files* formats vary from domain to domain, so the logic associated with the structural user interface display also varies based on the domain.

3.4.1 Structural User Interfaces Display:

At this stage, the framework-developer have to write the logic to display the appropriate user interfaces based on the content in the Resource-Listfile. For instance, in our case study, the Resource-Listfile (Format for App-Developer) as shown in the following Figure 11.

Figure 11: Resource-Listfile Format

The first row contains the information regarding the number of resources the application developer introduced. The rest of the rows contains the names of the resources. In this case, we wrote the logic (Chapter 4) to display the user interface that has the same number of buttons as of number of resources and the logic to populate the contents of the buttons with the resource name.

3.4.2 Behavioral User Interface Display:

In this section, the framework developer must write the logic to display the appropriate user interface based on the behavioral content in the "Data-and-Resource-File." For example, in our case study, we defined the behavior of the resource depends on Hours and Days. The information regarding the resource behavior is present inside the method (*getHorD())* as shown in Figure 12.

```
public class RoomReservation implements Resource {
  public String getHorD() {  return "H";  }   → // Resource Dependent Method
  public String[ ][ ] getResourceMatrix() {
      String[ ][ ] resourceGrid = { Data };
              return resourceGrid;    }
  }
```

Figure 12: Data-and-Behavioral-File

The framework-developer has to write the code to display appropriate user interface based on the contentment in the method getHorD(). In this case, we have two kinds of user interfaces they are displayed by the framework based on the following criteria: A) If getHorD() contains letter "H" then it will display the screen that accepts hours and B) If getHorD() contains letter "D" then it will display the screen that accepts hours. Both these cases are explained in the Section 4.4.2.

3.5 Summary

This chapter explains the systematic approach we are following to generate the domain-specific augmented framework. The approach consists of three steps: Graphical User Interface development, Application logic development, and User Interface logic. In the first step, Graphical User Interface development explains about the creation of skeleton user interfaces, which are categorized into Static and Dynamic User Interfaces. In the next step, we discuss the application logic development; it explains the dynamic integration of customization components into the framework. In the third step, we discuss the writing of the User Interface logic; its main function is to display the appropriate user interfaces based on the runtime input we get from the previous step.

CHAPTER 4

A CASE-STUDY FOR THE DEVELOPMENT OF AN AUGMENTED FRAMEWORK FOR MANAGING RESOURCES

In this chapter will show the development of augmented framework by following the steps mentioned in the previous chapter. Section 4.1 describes the domain to which we are applying the proposed steps. Sections 4.2 shows the development of layouts for the static and dynamic User-Interfaces that are associated with chose domain. Section 4.3 illustrates the dynamic binding process of Resource-Listfile as well as Data-and-Behavioral-Files. Section 4.4 explains the logic to vary the user-interface based on the input data.

4.1 Description of the Case-Study

To show the effectiveness of the proposed approach we are developing an augmented framework for resource allocation (Chosen Domain) for the reservation system. The augmented framework created by using the proposed approach can manage any resources associated with the reservation system. For instance, if we want to apply this framework (Reservation System) to libraries it will automatically handle different types of resources (i.e., Books, Study Rooms, Projectors, etc.) that are associated with the reservation.

4.2 User Interface Screens for Resource Allocation

As mentioned in Chapter 3 the user interfaces are divided into two types: Static and Dynamic. The following sections describe both types of user interfaces, as it relates to the case study.

4.2.1 Development of Static User Interfaces:

At this stage, the framework-developer need to develop the user interfaces that won't change based on the input. The Static User Interfaces are mostly common in many applications, such as the User Interfaces for authentication and admin functionality. We are developing the user interfaces as shown in the following figures.

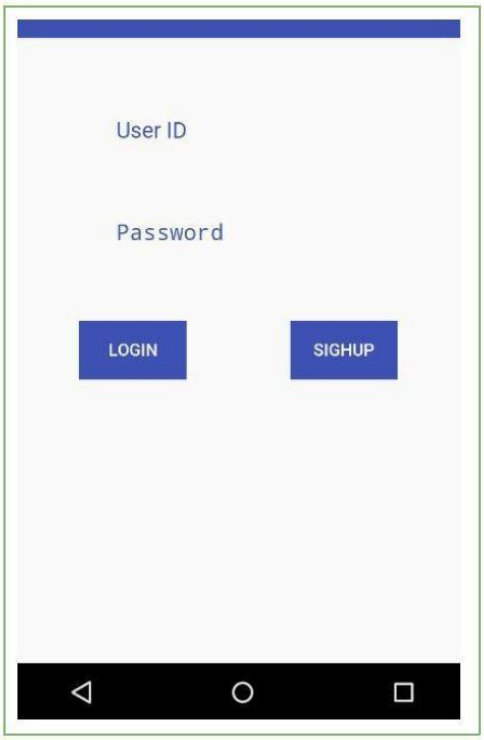


Figure 13: UI for Login Screen

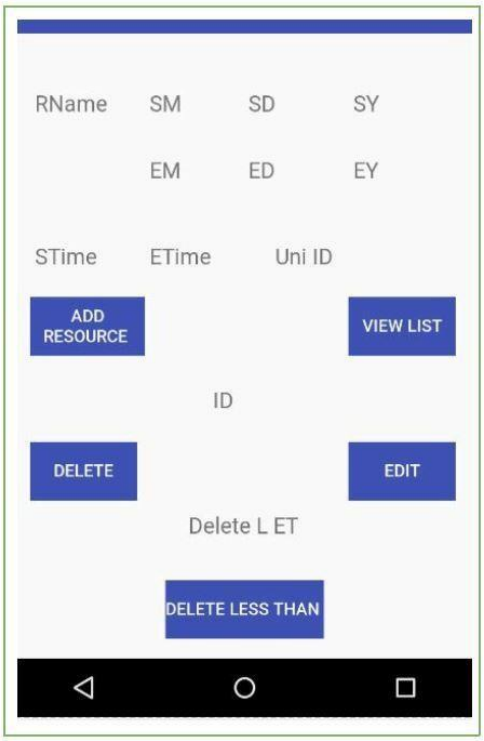


Figure 14: UI for Admin Screen

4.2.2 Development of Dynamic User Interfaces:

As mentioned in Chapter 3 Dynamic User Interfaces are dependent on the following factors:

1) Behavioral dependency.

2) Structural dependency.

Both are completely dependent on the domain we are selecting. As in our case study, we are selecting choosing the domain as the reservation system. The behavior in our case study is the duration, and the Structure is the number of resources the application is going to manage. The user-interface development of both categories is mentioned below.

4.2.2.1 Behavioral Dependency related to Duration factor:

If we look at the reservation system, the resources are Books, Projectors, and Study Rooms. They are directly dependent on the *Duration factor*, i.e., Days or Hours. So, we are categorizing the resources that are dependent on hours (Study Rooms, Conference Rooms, etc.) as one category and resources that are dependent on days (Books, Projectors, etc.) as one category. Then we will create the user interfaces that are commonly used for each category they are as shown in the following figures.

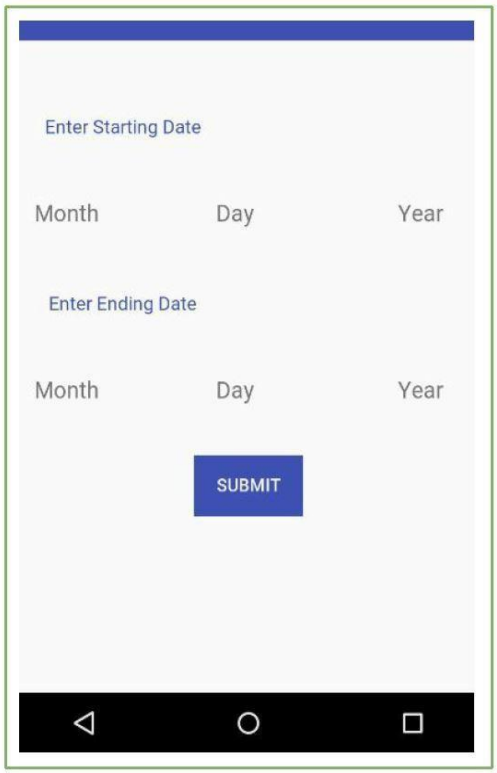


Figure 15: UI for Days

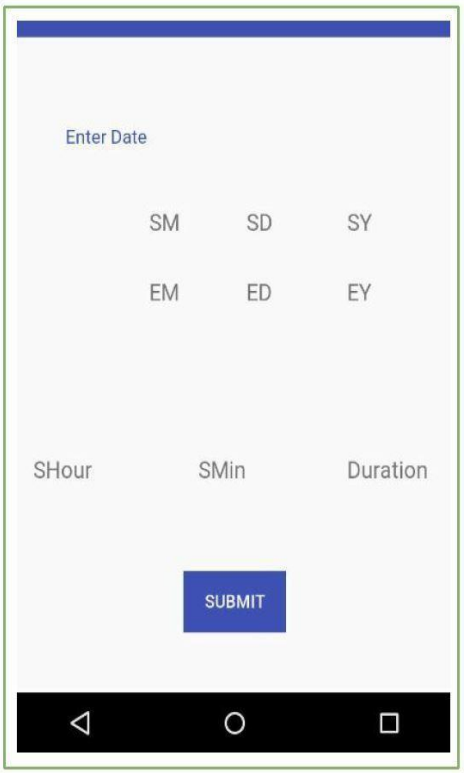


Figure 16: UI for Hours

Figure 13, will be displayed when the input resource depends on Days (Study Rooms, Conference Rooms, etc.). Figure 14, will be displayed when the input resource depends on Hours (Books, Projectors, etc.).

4.2.2.2 Structural Dependency related to Number of Inputs:

The structural dependency in our case study depends on the number of inputs the framework is going to manage. Since they are not known while creating the framework the framework developer must create the user interfaces that can accommodate any number of resources that comes during the runtime. For example, if there is only one

input the application display User Interface as shown in Figure 17 and for six inputs it should display Figure 18.

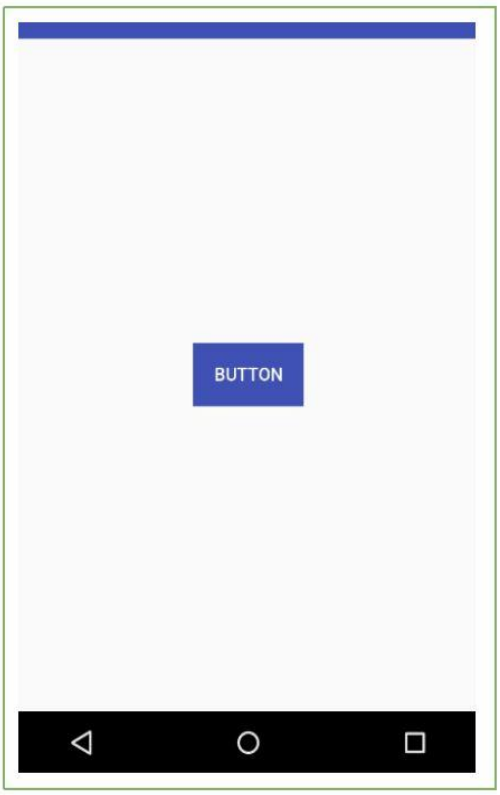


Figure 17: UI for One Resource

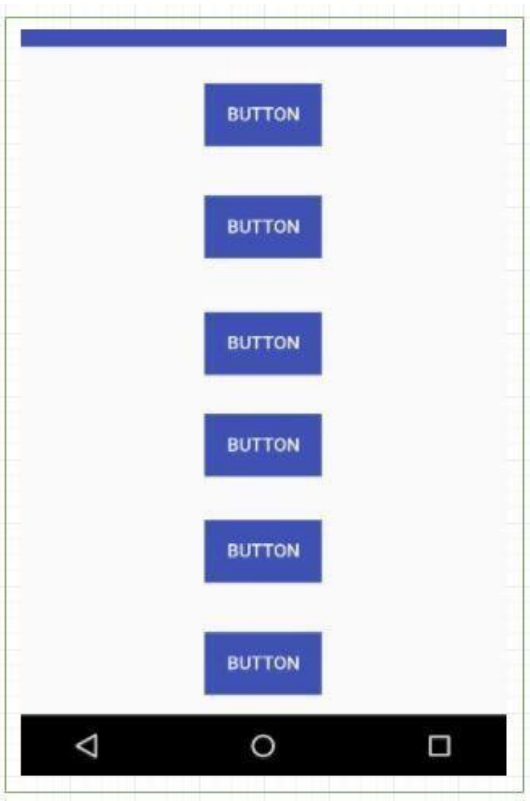


Figure 18: UI for Six Resources

Note: The content in the buttons are populated with the Resource names.

4.3 Application Logic for the Case-Study

This step is mainly associated with developing the mechanism to integrate the customization components into the framework, which we call it as dynamic binding. For as mentioned in the previous chapter it will accept two types of inputs Resource-Listfile and Data-and-Behavioral-Files. Resource-Listfile is useful to identify the resource and

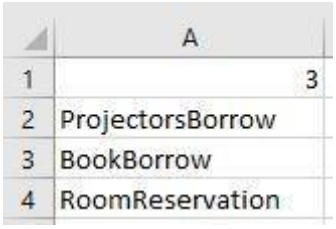
Data-and-Behavioral-Files are the set of files that contain the information regarding the resource. The formats and dynamic integration of each of them are explained in the following subsections. In this case study, *Resource-Listfile* is a *CSV File*, and *Data-and-Behavioral-Files* are *Java classes* both of them are discussed in the following sections subsections.

4.3.1 Source File for Identifying the Resources:

In our case study as a framework-developer, we are fixing the "*Resource-Listfile"* as the CSV File and the pattern as shown in Figure 19. The Structural dependent user-interface was selected based on the values in the first row (Number of Resources).

Figure 19: Resource-Listfile Format

Example of CSV Input file to the framework is as shown in Figure 8



	A
1	3
2	ProjectorsBorrow
3	BookBorrow
4	RoomReservation

Figure 20: Resource-Listfile Example

Integration of the CSV Files into the framework was done by using a simple code as shown in the following Figure 21:

```
InputStream inputStream;
inputStream = getResources().openRawResource(R.raw.resourcer);
BufferedReader reader = new BufferedReader(new InputStreamReader(inputStream));
```

Figure 21: Code to integrate the Resource-Listfile

4.3.2 Resource file with Behavioral Characteristics:

Data-and-Behavioral-Files are the files that contain resource information, i.e., data and its behavior. To follow a single input pattern as a framework-developer, we are mentioning the application-developer to implement an interface as shown in the following Figure 22 A. In the Interface, the method getHorD() defines the behavior, and the method getResourceMatrix() contains the complete data of the resource.

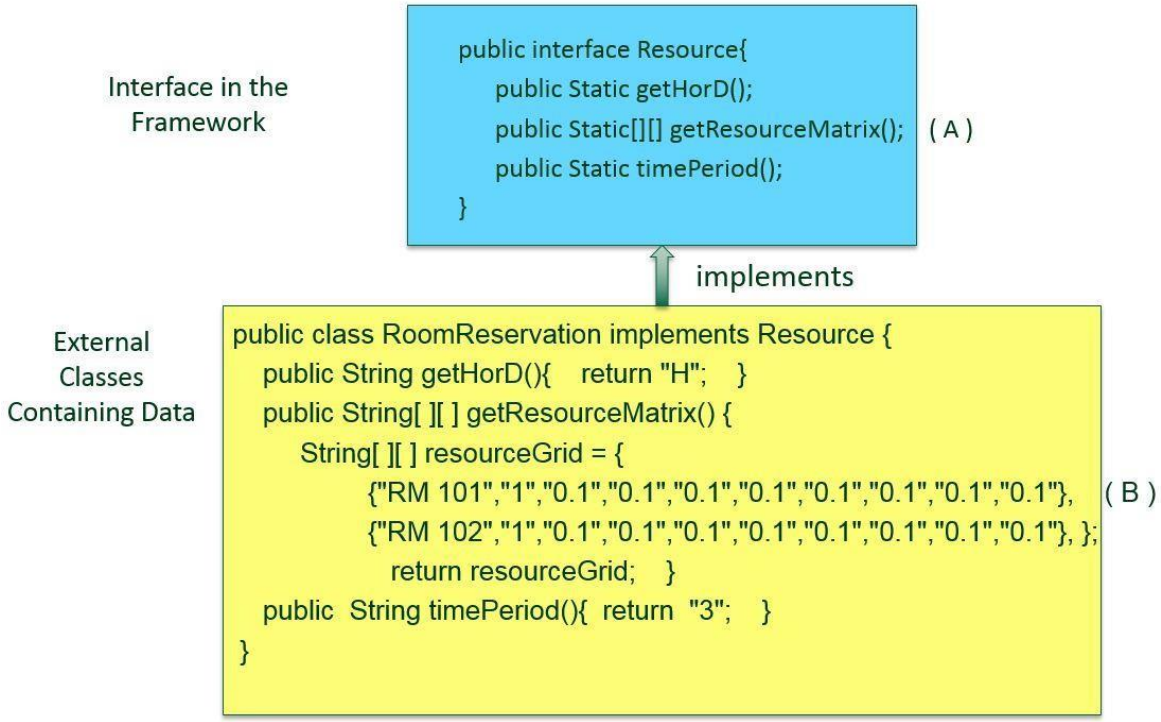


Figure 22: Fixed Input Convention

As an example, the Figure 22 B, shows the customization input given by the application-developer by implementing the interface. It contains both behavioral as well as data methods. If there is any mismatch in the input format, the framework won't accept the external files and indicates the application-developer.

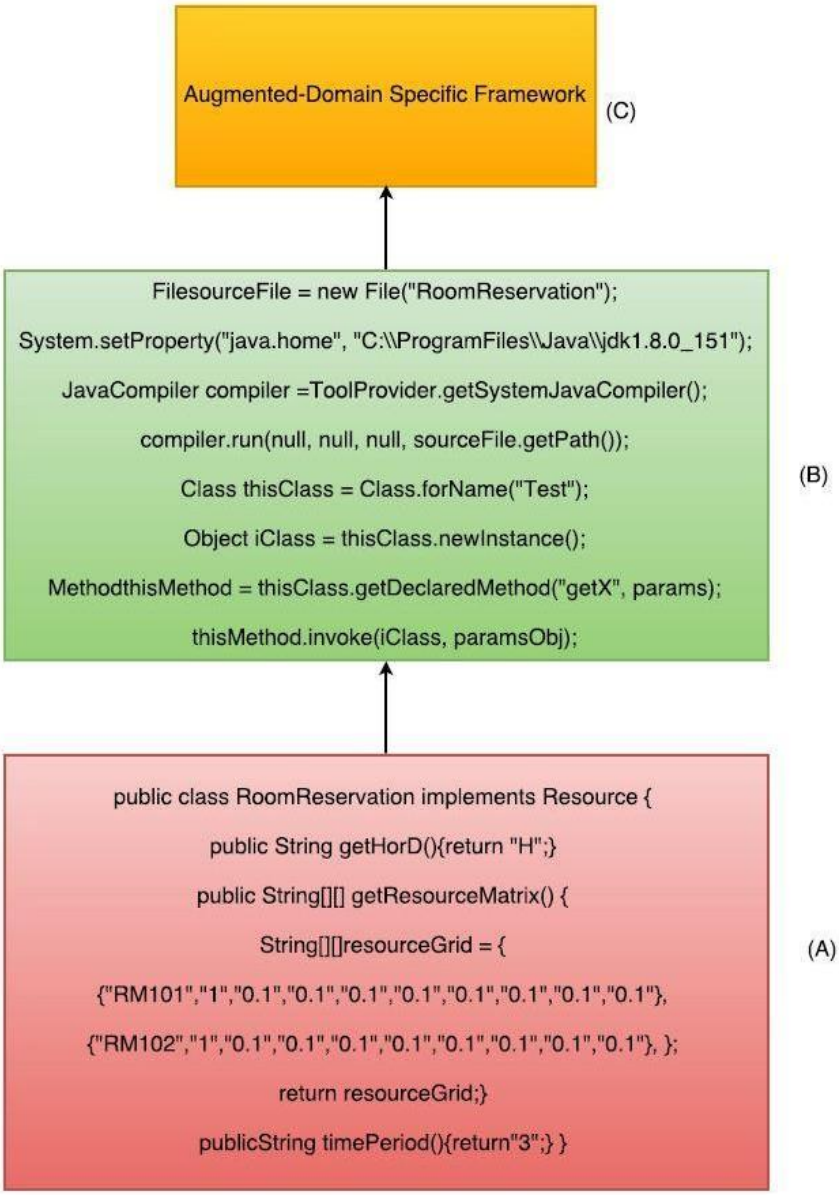


Figure 23: Process of Accepting Behavioral Files

Once the application-developer follows the input protocol format and inputs the text file as shown in Figure 23 A. The textfile will undergo the six-step process by going through the code as shown in the Figure 23 B. Once the input undergoes through the six-step process the input files are automatically integrated into the framework and all the methods in the input files are accessible to the Augmented Framework.

4.4 User-Interface Logic Development in our Case-Study

From the section 4.2, we got all kinds of user interfaces (Static and Dynamic) associated with the reservation system. From the section 4.3, the Augmented Framework has the mechanism to integrate the CSV Files (Resource Identification File) as well as the Java Classes (Resource Files). This stage consists of writing logic to generate the appropriate user interfaces based on the structural and behavioral data.

4.4.1 Generating Structural User Interface:

This step involves writing the code to Identifying the number of resource in CSV file and displaying the associated User Interface. From Section 4.3.1, we wrote the program to stored the CSV content in *reader* variable, by using the code as shown in Figure 24, we will get the number of resources, and it is stored in *resourceCount* variable.

```
if ((csvLine = reader.readLine()) != null) {
        ids = csvLine.split(",");
          resourceCount = ids[0];}
```

Figure 24: Code to Grab the Number of Resources

Based on the *resourceCount*, the pattern of displaying the user interface is as shown the following Figure 20.

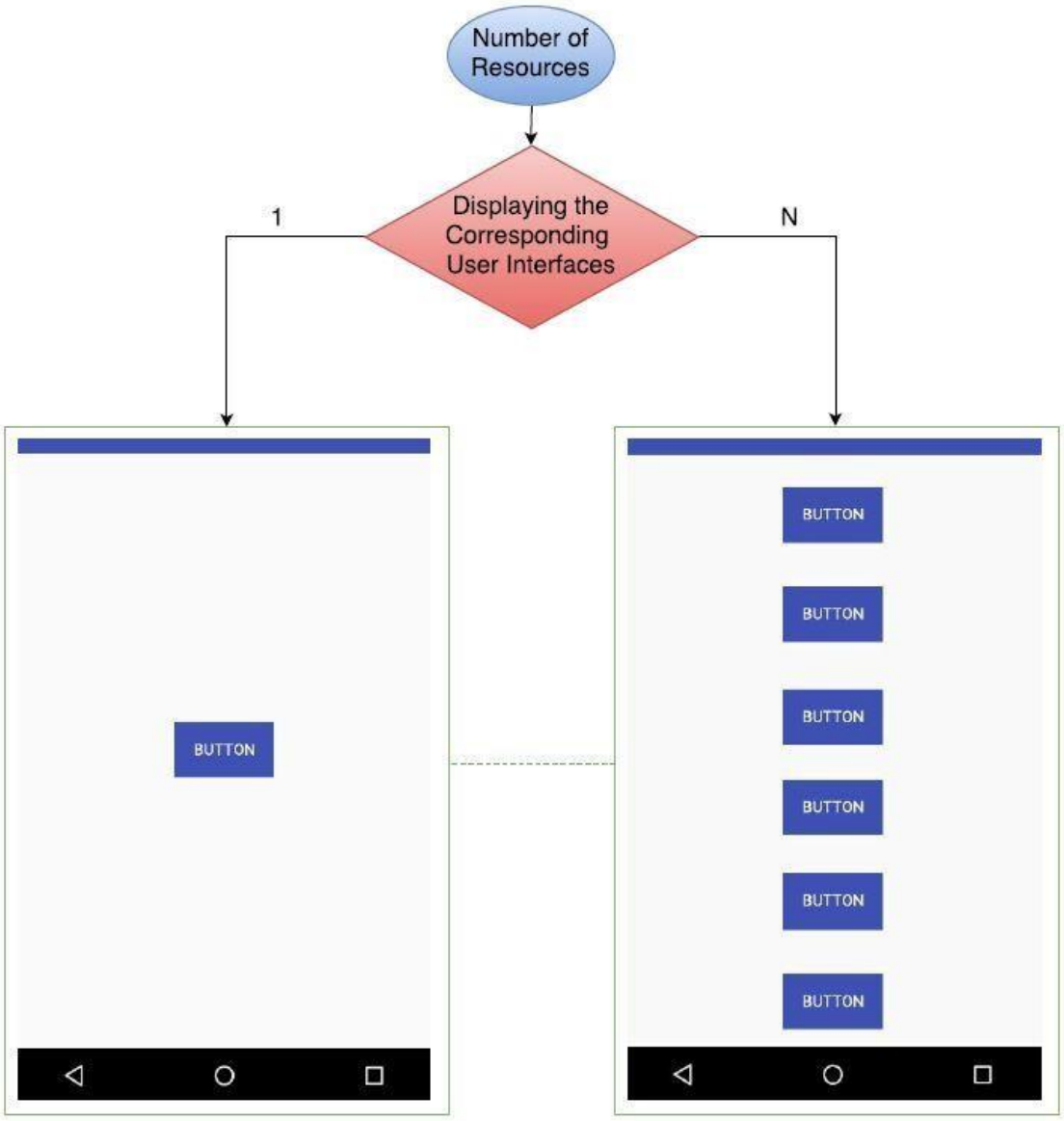


Figure 25: UI Display Based on Resource Count

CSV Files also contains the resource class names; They are used to identify the Resource Files. The code associated with identifying Resource File names from CSV File is shown in the Following Figure 26. In Figure 26, the array *classNames* contains all the resource file names. Once we get the list of class names, we have to go through each of the resource files to get the behavioral information.

```
while ((csvLine = reader.readLine()) != null)
{
     ids = csvLine.split(",");
     if (n == 0) {n++;}
     else if (n == 1)
     {
         className[i] = ids[0];
           i++;
     }
}
```

Figure 26: Code to Grab Resource File Names

4.4.2 Generating Behavioral User Interfaces:

This step involves writing the code to display User Interfaces based on the Behavioral data from the input classes. In our case study as mentioned in the 4.3.2, the behavioral components will come from the method "getHorD()." It has two possible values that are 'H' or 'D.'

```
Resource ob;
for (int j = 0; j < classCount; j++) {
   String inputJavaClass;
   inputJavaClass = className[j];

   String resName = "com.example.rrkr2016.javaclassname." + inputJavaClass;

      Class exampleClass = Class.forName(resName);
      ob = (Resource) exampleClass.newInstance();
      resourceGrid = ob.getResourceMatrix();

     resource = ob.getHorD();  // Extracting the Resource Behavior
```

Figure 27: Code to Extract the Resource Behavior

The above Figure 27, shows the code responsible for navigation towards the corresponding resource file and extracting the behavioral information from the resource

file. Once we get the content of the getHorD() method, the application will display the corresponding user interface. The pattern of displaying the User Interfaces based on Days or Hours is as shown in the following Figure 28.

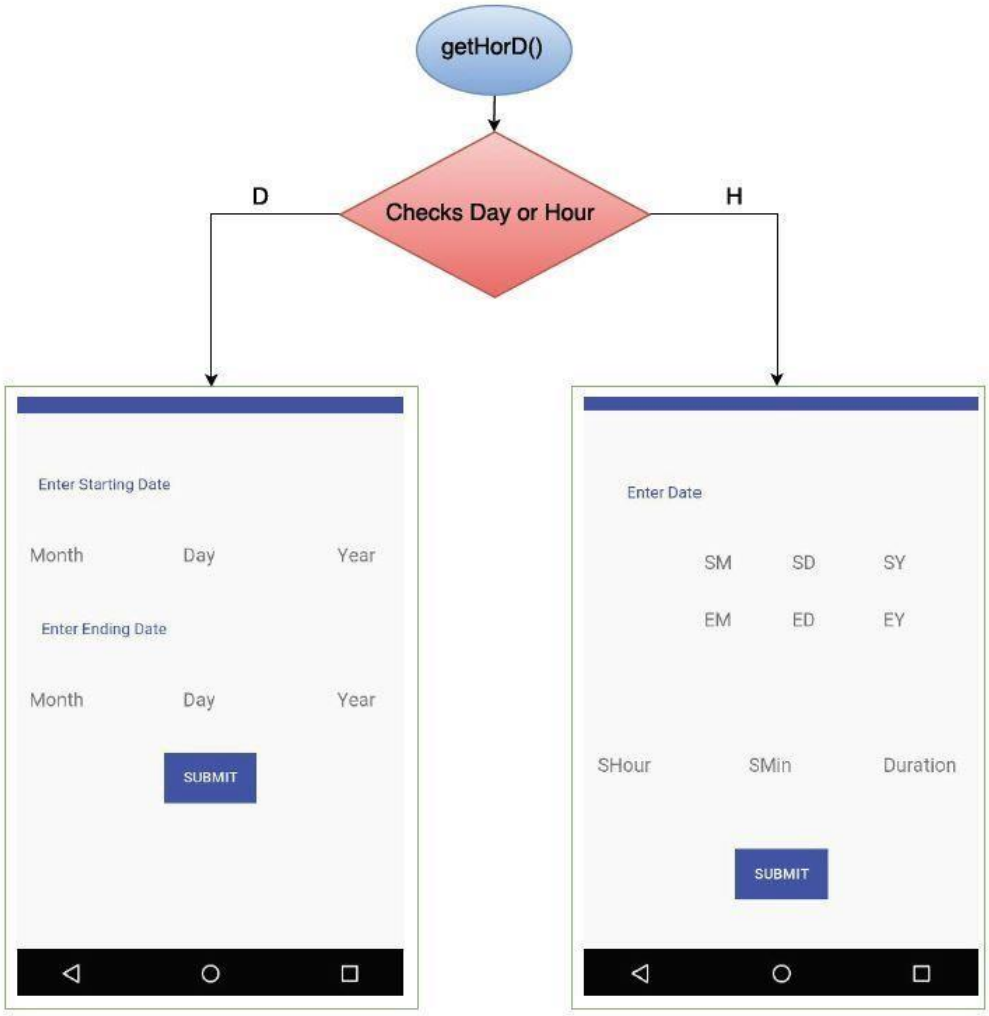


Figure 28: UI Display Based on Days or Hours

4.5 Summary

In this chapter, we demonstrate the effectiveness of the technology by using a case study. The chosen domain for our case study is a library reservation system. In the first step, we created examples of possible user interfaces used for the reservation of any

resources (Books, Study Rooms, Projectors, etc.). In the second stage, we used only one format of input and also wrote the code to integrate the customization components into the framework. In the third stage, we wrote the logic to display appropriate user interfaces based on the customization components at runtime.

CHAPTER 5

EVALUATION AND FUTURE WORK

In this chapter, we evaluate the limitations, summary and future scope of the proposed system. We compared the system with existing, similar frameworks. The chapter also highlights areas where the proposed approach can be improved. In Section 5.1, we evaluate the proposed approach by comparing it with the regular framework development process. Section 5.2 illustrates the limitations of the applications generated by the proposed approach. Section 5.3 briefly summarized our whole work. Section 5.4 gives an overview of the areas where the proposed approach can be extended.

5.1 Evaluation of Proposed Approach

Evaluation is necessary for any research to correlate with existing systems. In our research, we developed an approach for creating a new generation domain specific framework for mobile application development. There are many traditional mobile application development frameworks. Those frameworks follow the "Programming-in-the-large" model, focusing on simplifying the programming. However, the traditional frameworks require a deep understanding of multiple fields like scripting languages for user-interface, programming languages to generate dynamic effects, event handling, and SQL as well as RDBMS concepts for database operations. This makes the application development process complex and time-consuming. The proposed framework changes the

application development model to “Programming-in-the-Small” to reduce the application developer effort. We achieved it by integrating the domain knowledge with the traditional framework. The domain knowledge will help the application-developer to auto-generating the user interfaces. Nowadays we see so many frameworks which accept external data to develop customized applications. Developers follow the traditional approach of writing the code in the IDE provided by the framework. In our approach, we are not restricting application-developers to follow any developing environment; they can simply write the code in a “Text Document,” using the format specified by the framework-developer. Another advantage of this approach is in the inherent code reusability, which will contribute to a more reliable code since the framework will be thoroughly tested over time.

5.2 Limitations of Proposed Approach

Limitations do exist for every application development framework because no framework or development method can address all domains. The proposed system improves the application development process by adding the domain knowledge to the existing framework. On the other hand, we are restricting the framework for only the chosen-domain. If the application developer needs to develop the applications for different domains, he must search the other framework that associated with that domain. The pattern of the customization components may vary with the domains as well as framework-developer. Even though the proposed augmented framework allows customization up to some degree, but it is practically impossible to change the skeleton of the applications generated from the proposed framework. The proposed approach simplifies the application

development process, but on the other hand, it demands the very high expertise from the framework-developer.

5.3 Summary of Our Research

Nowadays there are many frameworks for the development of the mobile applications, but they still demand the programming expertise. To reduce this effort, we proposed an approach to extend the capabilities of a framework. The capability extension of a framework on the domain basis, i.e., adding the most commonly used features in the chosen domain into the framework by framework-developer. We call the framework obtained in this process as the Augmented Framework. Once the framework was developed the framework-developer will give a set of simple instructions on the external data acceptable pattern. The application-developer need to follow the set of instructions given by the framework-developer to give the customization components. After giving the customization components, the framework developer needs to run the augmented framework. The augmented framework will integrate the customization components into the framework and generate the application based on that input data. To further reduce the use of IDE's, we have developed a mechanism called "Dynamic-Binding" which will accept Text Documents.

5.4 Future Work

In general, frameworks always evolve over time. The following are the three main areas to extend our work:

1) Database Design Automation:

The proposed approach plays a significant role in auto-generating the user interfaces. As a next step, we can extend the auto-generating aspect to the database design, i.e., external data dependent databases.

2) Other Platforms:

The proposed approach was implemented for the Android platform, but it can be easily extendable to other Mobile platforms such as iOS.

3) Superior Methodology:

The proposed approach plays a good role in the development of mobile applications for resource allocation system. In the future, we will work on extending this research to provide a methodology, which can be applied to any domains.

LIST OF REFERENCES

[1] Frank DeRemer, Hans Kron, Programming-in-the large versus programming-in-the-small, proceedings of the international conference on Reliable software, p.114-121, April 21-23, 1975, Los Angeles, California.

[2] F.P Brooks Jr. No Silver Buller: Essence and Accidents of Software Engineering, IEEE Computer (April 1987), pp. 10-19

[3] https://dl.acm.org/citation.cfm ? id=1297846 & picked=prox & CFID=1013418917 & CFTOKEN=18720603

[4] Gregory F. Rogers, Framework-based software development in C++, Prentice-Hall, Inc., Upper Saddle River, NJ, 1997.

[5] https://search.proquest.com/docview/215837131?accountid=8240.

[6] M.R.J Qureshi, S.A.Hussain, A reusable software component-based development process model, Advances in Engineering Software, February 2008.

[7] F. Fdez-Riverola, D. Glez-Peña, H. López-Fernández, J.R.A. Méndez Java application framework for scientific software development. Software: Practice and Experience (2012).

[8] J. Greenfield, and K. Short; Software Factories: Assembling Applications with Patterns, Models, Frameworks and Tolls, John Wiley & Sons, 2004.

[9] Ponder M, Papagiannakis G, Molet T, Magnenat-Thalmann N, Thalmann D. VHD++ development framework: towards extendible, component based VR/AR simulation engine featuring advanced virtual character technologies. In Proceedings of Computer Graphics International (CGI). IEEE Computer Society Press, 2003.

[10] S. A. Amjad and S. A. Khan, "A Framework for Enhancing Readability and Opportunistic Reuse of Enterprise Software," New York, 2015.

[11] http://vhdplus.sourceforge.net/vhdPlusDoc.html

[12] W.W. Agresti, F.E. McGarry, "The Minnowbrook Workshop on Software Reuse: A Summary Report," *Tutorial Software Reuse: Emerging Technology*, 1987.

[13] J.J. Jeng, B.C.H. Cheng, "Specification Matching for Software Reuse: A Foundation."

[14] Zhu, F., Turner, M., Kotsipoulos, I.A., Bennett, K.H. (2004). Dynamic data integration using web services.

[15] A. Schultz, A. Matteini, R. Isele, C. Bizer, and C. Becker. LDIF - Linked Data Integration Framework, 2011.

[16] A framework for effective commercial Web application development, Research paper Ming-te Lu, Wing-Lok Yeung, 1998.

[17] M. Lenzerini. Data integration: A theoretical perspective. In *Proc. PODS'02*, pages 233–246. ACM, 2002.

[18] V. Basili, L. Briand, and W. Melo. How reuse influences productivity in object-oriented systems. Communications of the ACM, 1996.

[19] J.S. Poulin- Measuring Software Reusability, Proc. Third Conf Software Reuse, Nov. 1994.

[20] Martin L. Griss, Software reuse: from library to factory, IBM Systems Journal, v.32 n.4, p.548-566, October 1993.