DYNAMIC MULTI-TARGET USER INTERFACE DESIGN FOR GRID BASED M-SERVICES

Ipadeola Abayomi .O

(200711858)

A dissertation submitted in fulfillment of the requirements for the degree of

Masters of Science in Computer Science

Department of Computer Science, Faculty of Science and Agriculture, University of Zululand.

South Africa

2008

Declaration

This dissertation represents the author's own research work and has not been submitted in any form to any other tertiary education institution for another degree or diploma. All the material that has been used as source of information has been acknowledged in the text.

Signature

Dedication

To God almighty, the giver of life and the creator of all mankind

Acknowledgements

I acknowledge my source, the giver of life, the one and only God and my greatest shepherd whose divine assistance is unquantifiable and has successfully paddled me this far. My sincere gratitude goes to my supervisor, Dr S.S. Xulu and my co-supervisor Prof M.O. Adigun for their tutelage and contribution towards the success of this work. Much appreciation also goes to my sponsors, members of staff of Computer Science Department and co-researchers in the Centre of Excellence, University of Zululand, South Africa.

A deep sense of appreciation also goes to senior colleagues, Dr D.R. Aremu, Mr J.S. Iyilade, Mr E. Jembere, and Mr Ekabua. I will specially acknowledge Dr O.O. Olugbara and Dr Kwake, for their unquantifiable support during the course of this study. Your guidance, encouragement and admonitions can never be over-emphasized. I appreciate my loving parents, Engr and Mrs B.M Ipadeola and my brothers, Yinka, Ladi and Laolu. You are indeed irreplaceable. Your care, love and prayers mean so much to me.

I appreciate Toyin and Tobi Oluwafemi for their love and care. I specially acknowledge Miss Mercy, O. Bunmi for her support during the course of this study. You are indeed a rare gem. Similarly, I appreciate the members of University of Zululand Christian Fellowship (UZCF). The arms of love and acceptance you showed me are unquantifiable.

Finally, I give all glory and all adoration to the Lord Jesus Christ, for being a friend and a father. With you, I am never alone.

Table of Contents

CHAPTER ONE 1
INTRODUCTION 1
1.1 Background 1

1.2 Statement of the Problem 3

1.3 Rationale of the Study 4

1.4 Goal and Objectives 5

1.4.1 Research Goal 5
1.4.2 Research Objectives 5
1.5. Significance of the Study 5

1.6 Research Methodology 7

1.6.1 Literature Survey 7
1.6.2 Algorithm Formulation 8
1.6.3 Proof of Concept Approach 8
1.7 Organization of the Dissertation 8

1.8 Summary ..... **Error! Bookmark not defined.**

CHAPTER TWO 11
LITERATURE REVIEW 11
2.1 Introduction 11

2.2 User Interfaces for Personal Communication Devices 13

2.3 Mobile Applications and Services Requiring User Interfaces 16

2.3.1 Web applications 16
2.3.2 Web Service 17
2.3.3 Service Grid 17
2.4 User Interface Classes 18

2.4.1 Multi-Device Interfaces 18
2.4.2 Multi- Environment User interface 19
2.4.3 Multi-Target User Interface 19
2.5 User Interface Adaptation Approaches 19

2.5.1 Middleware Based Adaptation Approach 20
2.5.2 The Model Based Approach for User Interface Design 20

2.5.3 Mark-up Languages Adaptation 24
2.6 Model Based Approaches to Multi-target User Interface Design. 24

2.6.1 Device Centered User Interfaces 25
2.6.2 User Centered Interfaces 29
2.7 The Java 2 Micro Edition Platforms and the Browser Based Environments.... 34

2.7.1 Advantages of J2ME Platform over the Browser Based Environment..... 34
2.7.2 Adaptable Interfaces for J2ME Enabled Devices. 36
2.8 Summary .....**Error! Bookmark not defined.**

CHAPTER THREE 38
MODEL DEVELOPMENT 38
3.1 Introduction 38

3.2 The Polymorphic Logical Description (PLD) MODEL 39

3.2.1 Polymorphic Task Modeling (PTM) 41
3.2.2 Polymorphic Abstract Modeling Phase 44
3.2.3 Polymorphic Concrete Modeling Phase 45
3.3 Architecture for Integrating PLD into a Mobile Computing Environment

(iPLD). 47

3.4 Load and Run Time Tiers of iPLD 50

3.5 The iPLD Repositories 51

3.5.1 Concrete Object Repository 52
3.5.2 Active User Repository 52
3.5.3 Media Content Repository 52
3.6 User Request/Response Information Communication 53

3.6.1 User Information Bus (UIB) 53
3.6.2 An Interface Shopping Cart for Preference Handling 53
3.7 Core Adaptation Components 56

3.7.1 Interface Adaptation Engine 56
3.7.2 Content Adaptation Engine (CAE) 58
3.8 Packager 59

3.9 Modeling Algorithms for the Proposed Method 59

3.9.1 Algorithm for creating Polymorphic Task Model using the CoMADE toolkit 59

3.9.2 Algorithm for transforming Polymorphic Task Model to Polymorphic Abstract Model 61
3.9.3 Algorithm for transforming Polymorphic Abstract Model to Polymorphic Concrete Model 62
3.9.4 Algorithm for transforming Polymorphic Concrete Model to Final User Interfaces 63
3.10 Summary 65
CHAPTER FOUR 66
PROTOTYPE DESIGN, IMPLEMENTATION AND EVALUATION 66
4.1 Introduction 66
4.2 Prototype Design 67
4.2.1 An Overview of the toolkit 67
4.2.2 Design Time Activities 68
4.1.3 The Load and Runtime activities 69
4.3 The Implementation of CoMADE Toolkit 71
4.4 The CoMADE Environment 75
4.4.1 A Polymorphic Task Modeling Section 76
4.4.2 A Polymorphic Abstract Modeling Section 78
4.4.3 Polymorphic Concrete Modeling and Final User Interface Generation 80
4.4.4 The Mediator 84
4.5 The Evaluation of the Proposed System 85
4.5.1 Evaluation Background and Logistics 86
4.5.2 Evaluation Procedure 87
4.5.3 The CoMADE Generated User Interfaces 88
4.5.4 Evaluation Results 89
4.6 Discussion of the Result 97
4.7 Summary 98
CHAPTER FIVE 99
CONCLUSION AND FUTURE WORK 99
5.0 Introduction 99
5.1 Summary of Contribution 100
5.2 Conclusion 105
5.3 Future Work 106
APPENDIX A 120
APPENDIX B 124
APPENDIX C 127

List of Figures

Figure 1. 1 The GUISET Architecture 7
Figure 2. 1:Three different devices with the same user interface presentation 1
Figure 2. 2: Automatic Translation Method 1
Figure 2.3: Multi-Target Scenario. 15
Figure 2. 4: Sample Task Model for a GUISET User Interface 1
Figure 2. 5: Concrete User Interface Description 1
Figure 2. 6: Snapshot of DAMASK Authoring Tool 26
Figure 2. 7: Presentation Structure Selection Using a Mediator 27
Figure 2. 8: Polymorphic Task Decomposition. 1
Figure 2. 9: Hierarchical Decomposition. 1
Figure 3. 1: Polymorphic Task Representation 42
Figure 3. 2: Polymorphic Abstract Interface Representation 1
Figure 3. 3: Polymorphic Concrete Model Persisted Into Db4o 1
Figure 3. 4 User Centered Process Model for Integrating PLD (iPLD) into a Mobile Computing Environment 49
Figure 3. 5 Load and Run time Tiers 1
Figure 3. 6: Real-Time Preference Submission 1
Figure 3. 7: Interface Composition Process 58
Figure 4. 1: Use Case of the Proposed System 68
Figure 4. 2: Sequence Diagram for the Creation of a PLD by an interface Designer 69
Figure 4. 3: Sequence Diagram for the Generation of Customized Interface by a User .. 70
Figure 4. 4: Activity Diagram for the Proposed System 1
Figure 4. 5: Required Implementation Steps of CoMADE 73
Figure 4. 6: Code for Dynamic Compilation and Pre-Verification by the Packager Component of the Mediator 74
Figure 4. 7: Interface Resource File 75
Figure 4. 8: The Polymorphic Task Modeling Section of CoMADE. 77
Figure 4. 9: The Polymorphic Abstract Modeling Section of CoMADE 78
Figure 4. 10 : Hierarchical Representation of Presentation Set Description Language.... 79

Figure 4. 11: Polymorphic Concrete Modeling: The Designer Work Board (A) 81
Figure 4. 12: Polymorphic Concrete Modeling: Interface Composition (B) 82
Figure 4. 13: Polymorphic Concrete Modeling: Explicit Compilation and Emulator Invocation by Designer 83
Figure 4. 14: The Polymorphic Concrete Modeling: Automatic Interface Generation by Mediator 84
Figure 4.15: The Mediator Integrated Into CoMADE 85
Figure 4.17: User's Choice - Lang: English, Image: grayed, Report: tables 1
Figure 4. 16: User's Choice - Lang: English, Image: Coloured , Report: Tables 1
Figure 4. 18: User's Choice - Lang: isiZulu, ... 1
Figure 4. 19: Evaluation of CoMADE's ability for the design of Polymorphic task model 90
Figure 4. 20: Evaluation of CoMADE's ability in automatic generation of Polymorphic abstract model 91
Figure 4. 21: Evaluation for CoMADE's Ability for Automatic Generation of Polymorphic Concrete Model 92
Figure 4. 22: CoMADE's ability for automatic generation of Final User Interface 93
Figure 4. 23: Evaluation of CoMADE in the generation of an interface based on specified preference 94
Figure 4. 24: Evaluation of CoMADE effectiveness in adapting an already generated interface to changes in preferences 95
Figure 4. 25: Response to the speed of CoMADE in multi-target user interface design .. 96
Figure 4. 26: Response to ease of use 97

List of Tables

Table 2. 1: Analysis of Contributions in Multi-Target User Interface Design 31

Table 5. 1: Comparative Analysis of Model-Based Approaches in User Interface Design 105

Abstract

Device heterogeneity and the diversity in user preferences are challenges, which must be addressed in order to achieve effective information communication in a mobile computing environment. Many model based approaches for the generation of user interfaces have been proposed. These approaches were aimed at achieving effective information communication in a mobile computing environment. Unfortunately, current model based approaches do not support user participation in interface adaptation. Hence, user interfaces are adapted for and on behalf of users, leaving users with inappropriate interfaces. There is therefore a need for an approach that allows direct user participation in interface adaptation. Such an approach must be responsive to changes in user needs and preferences. This work therefore proposes a Polymorphic Logical Description (PLD) method for automatic generation of multi-target user interfaces. The PLD method consists of user preference aware models, which are created at design time. Interface artifacts are considered in the PLD as methods with polymorphic attributes in a bid to address the diversities experienced in a mobile computing environment. An interface artifact is dynamically selected for inclusion on an interface based on the context information of a requesting user and intrinsic characteristics of a device. Our approach is enabled by a support toolkit, named Custom-MADE (CoMADE) for automatic interface generation. The evaluation result of CoMADE shows that it achieves a high degree of user participation during interface generation, flexibility, dynamism, ease of application extensibility and user centeredness interfaces.

CHAPTER ONE

INTRODUCTION

1.1 Background

Mobile computing can be described as a technology that allows transmission of data through a computer without having to be connected to a fixed physical link (Koudouna, and Omar 1996). This technology enables a mobile computing device to remain in virtual continuous contact with network resources. The advent of mobile computing has resulted in an ubiquitous style of information communication. The information access and transmission can now be carried out at anytime and anywhere. Important instruments for mobile information communication, known as Personal Communication Devices (PCD), include Personal Digital Assistants (PDA), Internet Enabled Televisions (WebTV), and Pagers (Vincent, 2002; Genco, et al., 2006). Information communication through these devices is associated with challenges, some of which are given below. These challenges are inherent in any typical mobile computing environment (Forman and John, 1994; Tao, et al., 2005).

Firstly, mobile computing devices are heterogeneous in nature. For example, PCDs have vastly different screen sizes, computing power, and storage capabilities. Secondly, these devices are characterized by limited resources such as small memory size, and low network bandwidth allocation (Brewster, et al., 1998; Olsen, et al., 1998; Richter, 2005). Another dimension of the challenges of a mobile computing environment

is the heterogeneity of users. Users have diverse needs, which can be defined in terms of their demographics, information seeking strategies, and purchasing intent factors (Burns, et al. 2001). Thus users are bound to have different preferences, different choices of interaction modality etc. Furthermore some users, for instance, users with disabilities, require special consideration (Aaron, 2003). One way of addressing the heterogeneities of computing devices and diversities in preferences and requirements of individual users is through the design of a separate user interface for each and every user device. According to Kai, (2005), this conventional method is considered inefficient due to the following reasons: It is time consuming and error-prone; it produces inconsistent interfaces across diverse devices, since an interface design for each device needs to be handled differently; in addition, this approach is not cost-effective.

In the light of these disadvantages, an automatic translation method was proposed (Barbara, 2007), which involves the creation of a generic logic design that is automatically adapted to diverse devices. A multi-target user interface is an automatic translation method that addresses the contextual information about the device (platform), the user of the device and the execution environment during user interface design. In essence, a multi-target interface supports multiple types of devices, users, and environments (David, et al., 2001).

Much work in multi-target user interface design has concentrated only on adapting an interface to multiple types of devices (Lonczewski and Shreifer, 1996; Mori, et al., 2004a; Kai, 2005). The adaptation of an interface to a user is currently performed for and on behalf of a user without the user participation. It involves hard coding of the requirements for adaptation (Bisignano, et al., 2006). This method hampers user

satisfaction and can leave users with unusable interfaces (Germanakos, et al., 2006). User involvement must, therefore, be given utmost consideration during user interface adaptation. In order to achieve universal usability of interfaces, users must be allowed to determine the nature of their interface presentation at a given time. The degree of user participation can be viewed from the following dimensions:

i. Ability of a user to determine the nature of an interface presentation through requirement specification.

ii. Ability of a user to alter an interface presentation by changing his or her user preference information.

iii. Ability of a user to view existing interface presentation structures and also generate variants of interface presentation structures at will.

iv. Ability to perform interface adaptation at both the design and runtime phases.

1.2 Statement of the Problem

Adapting user interfaces while hard coding the requirements for adaptation has left users with inefficient interfaces. User interface adaptation without user involvement has proved to be inefficient. To realize universally usable interfaces, direct user participation in user interface adaptation must be achieved.

The following questions, therefore, need to be answered. Can direct user participation be achieved in user interface adaptation without hard coding the

requirements for adaptation? How can users be allowed to customize their interfaces in terms of accessibility to information? What kinds of methods are needed for designing an interface for diverse users (Seffah, et al. 2004)?

1.3 Rationale of the Study

As the world goes pervasive and computing assumes everywhere, any time status, with solutions and user applications running on heterogeneous devices owned by diverse users, there arises the need for a dynamic approach that adapts interfaces to suit a user context. An approach, which gives priority to user participation during adaptation, will allow users to derive maximum satisfaction from using application interfaces.

The proposed method will, for instance, enable a tourist user to define, set or pre-set his or her location based-interface to suit her interest. A service provider will be able to monitor the performance of his/her services using a customized interface. Even designers will be able to automatically generate interfaces at will with little or no programming experience.

Hence, this approach will go a long way in enhancing the experience of users of solutions in e-Commerce, e-Business, e-Learning, e-Health, e-Tourism, etc. as they are able to vary the composition and usability of their interface at a given time.

1.4 Goal and Objectives

1.4.1 Research Goal

The goal of this research was to develop user interface modeling algorithms and toolkit for the design of multi-target interfaces that support direct user participation in the composition and adaptation of interfaces.

1.4.2 Research Objectives

The specific objectives of this research work are:

i. Evolve a model for achieving direct user participation in user interface composition and adaptation.

ii. Construct algorithms for user interface modelling and selection of sets of presentation structures suitable for users at a particular time.

iii. Develop a framework that provides infrastructures for integrating the proposed model into a mobile computing environment.

iv. Prototype a toolkit based on the proposed framework in (iii).

1.5. Significance of the Study

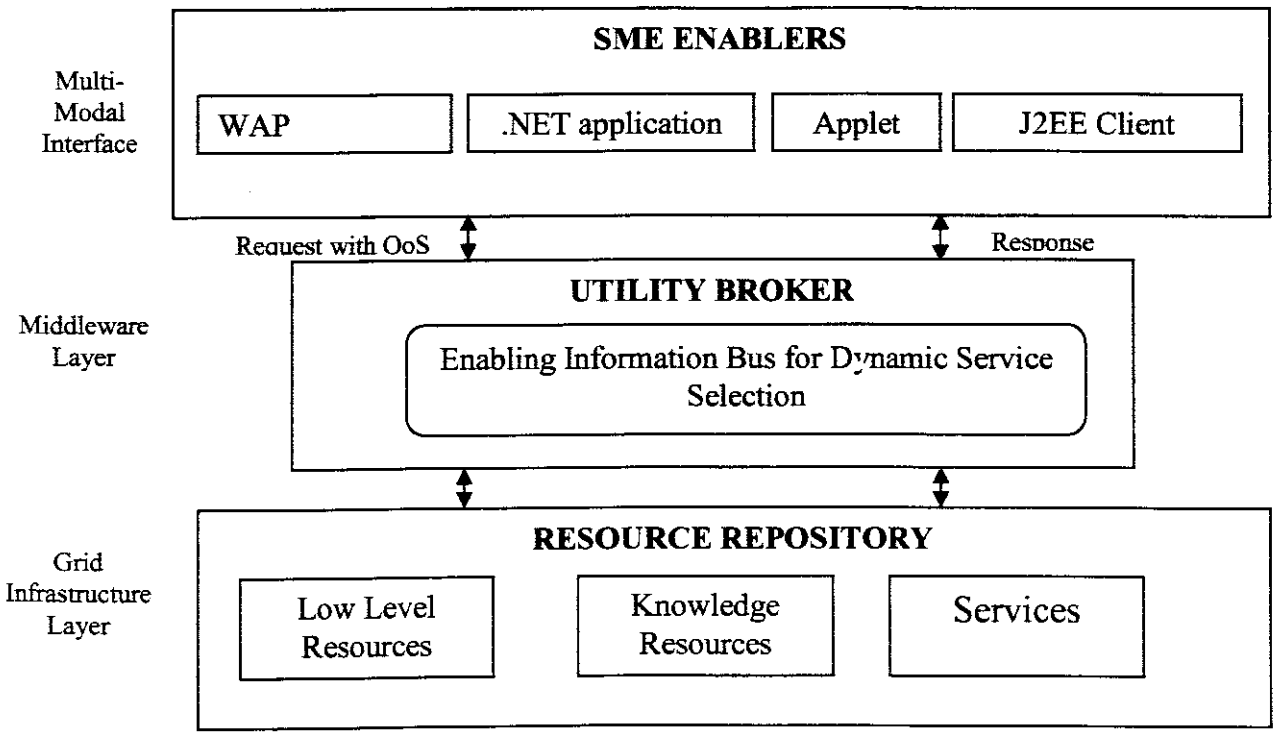
The significance of this study lies in the provisioning of multi-target user interfaces for access to services in Grid-based Utility Infrastructure for Small, Medium and Micro Enterprise (SMME) Enabling Technology (GUISET) (Adigun, 2006). GUISET is a

scientific apparatus created to facilitate the study of application and service components in a Mobile Grid Environment (MGE). GUISET initiative, being the niche research project of the Department in which this work was carried out, has the goal of empowering rural communities with a technology that can boost their social and economic status.

GUISET is a three-tier layer architecture comprising the Grid-infrastructure layer, middleware layer and multi-modal interface layer (Figure 2.4). The Grid infrastructure layer provides services and resources, which are managed by the middleware infrastructure. The Grid infrastructure layer provides a service user with repository of business support services, which are accessible on demand with the aid of an interface. The middleware infrastructure is made up of the utility broker, which enables services to be consumed as utility services. The multimodal interface layer provides support for universal accessibility to all services irrespective of device type, user diversity and execution environments. The aim of this work is to achieve this multi-modal layer by providing consumers of GUISET services with universally accessible interfaces.

Our approach to universal accessibility to services in a GUISET environment is direct user participation in multi-target user interface generation. In this study, a Polymorphic Logical Description (PLD) model was proposed and a support toolkit to aid designers in the specification of polymorphic interface descriptions at design time was developed to facilitate the generation of multi-target interfaces through direct user participation and address the challenges posed by diversity in the intrinsic characteristics of users, devices and GUISET executing environments.

Figure 1. 1 The GUISET Architecture



1.6 Research Methodology

The research approach employed in this study was two-fold in nature as both theoretical and formulative aspects were involved. Also, a number of research methodologies were considered in answering the research challenges raised in this study. They are as follows:

1.6.1 Literature Survey

An extensive literature review provided the basis for developing an appreciation of the relevant issues to the study. The knowledge gained from the literature survey was used to construct a theoretical background of the research. Different approaches for the design of multi-target user interfaces were investigated. In addition, this survey included

identifying commonly accepted approaches and methodologies from similar work, which formed the basis for proving the credibility of the model.

1.6.2 Algorithm Formulation

The formulative aspect of this research explored critical evaluative and comparative analysis of existing related approaches in multi-target user interface design. The analysis result was used to formulate the algorithms and tools, which contributed towards the achievement of the objectives of this study.

1.6.3 Proof of Concept Approach

The proof of concept involved prototyping of the proposed toolkit, implementation and evaluation of the toolkit. The toolkit demonstrated how direct user participation can be achieved in multi-target user interface design. The toolkit also shows how priority can be given to user preferences during multi-target user interface generation. An evaluation of the toolkit and the usability of the generated interfaces were carried out based on the commonly accepted approaches and methodologies identified in the literature survey (Nielsen, 2000).

1.7 Organization of the Dissertation

This dissertation is organized as follows:

Chapter one gives a brief background to mobile computing and the challenges facing effective information communication in a mobile computing environment. The

chapter gives the need for direct user participation in a multi-target user interface design. Furthermore, the chapter covers the purpose of this study.

Chapter two presents the literature review in user interface design. It specifically describes the various approaches that have been employed for achieving information communication through mobile devices. A description of some applications and services, which require interface design for end user accessibility, is given. In addition, three major classes of user interfaces are presented. The chapter reviews user interface adaptation approaches that can be employed for the design of any class of interface. In addition, existing model based approaches in user interface design were reviewed in relation to multi-target interface capability. Much emphasis is placed on the consideration of user's interest through user participation during multi-target user interface design.

Chapter three presents our proposed solution to direct user participation challenge of multi-target user interface design. It gives a description of our proposed model, framework and user interface modeling algorithms developed for achieving direct user participation during an interface design.

Chapter four explains the prototype design, implementation and evaluation of our toolkit Custom-MADE (CoMADE) for automatic generation of multi-target user interface. CoMADE is developed based on our proposed framework.

Chapter five gives the summary of the contribution of this work. It further gives a conclusion and some directions for future work.

1.8 Summary

The universal usability of user interfaces can be largely achieved when interfaces are generated according to specified user preference information. In multi-target user interface design, user interface generation without user involvement has been found to be inefficient. This study therefore, focuses on the review of existing methods in multi-target user interface design, description of a proposed model and implementation of a support toolkit, named CoMADE for achieving direct user participation during automatic multi-target interface generation.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

With recent advances in mobile technology, came the challenge of how mobile devices can be effectively used to gain ubiquitous access to information, services and applications. These challenges emanate from the fact that mobile devices are resource-constrained. Proponents of the mobile grid believe that techniques emanating from the research are enabling the infrastructures to support resource-constrained mobile devices for information, services and application accessibility (David, et al, 2005; Tao, et al, 2005; Isaiadis and Getov, 2005). However, handling the heterogeneities of the mobile computing environments, which include diversity in users, devices and environments is posing a challenge to universal accessibility to such services. We are of the opinion that universal accessibility to services and applications can be largely achieved through multi-target user interfaces.

An interface refers to a communication boundary between two entities. For example, an interface is a communication boundary between an end user and an artifact. The design of interfaces for mobile devices requires special consideration due to their peculiar characteristics. The design of user interfaces should be skillfully done to aid end user interaction with mobile applications, services and for information accessibility in general. A well designed user interface is, therefore, vital for effective information communication in a mobile computing environment. Some examples of mobile

computing devices include mobile phones, pagers, WebTV and PDA. This chapter presents some background concepts and related work in user interface design for mobile computing devices.

In section 2.2, we consider the conventional method of achieving user interface presentation for diverse PCDs. Software systems, which include mobile applications and services requiring user interface design for user accessibility, are presented in Section 2.3. Mobile applications and services are presented to users through different classes of interfaces. Section 2.4, therefore, describes the various classes of user interfaces that can be designed to aid accessibility to application and services. In Section 2.5 we discuss the various adaptation approaches for achieving different classes of interfaces. Finally section 2.6 gives an analysis of existing model based approaches in multi-target user interface design. A user interface can be categorized into two kinds, namely, the browser based and the device-based interfaces. The browser based interfaces are hosted on a web or Wireless Application Protocol (WAP) servers and are accessible through a client device while interfaces generated with the J2ME platform are referred to as device-based interfaces. Device based interfaces are downloaded onto a client's device where it permanently or temporarily resides. Section 2.7 therefore compares the Java 2 Micro Edition (J2ME) platforms and the browser based environment. Section 2.8 gives the conclusion to this chapter.

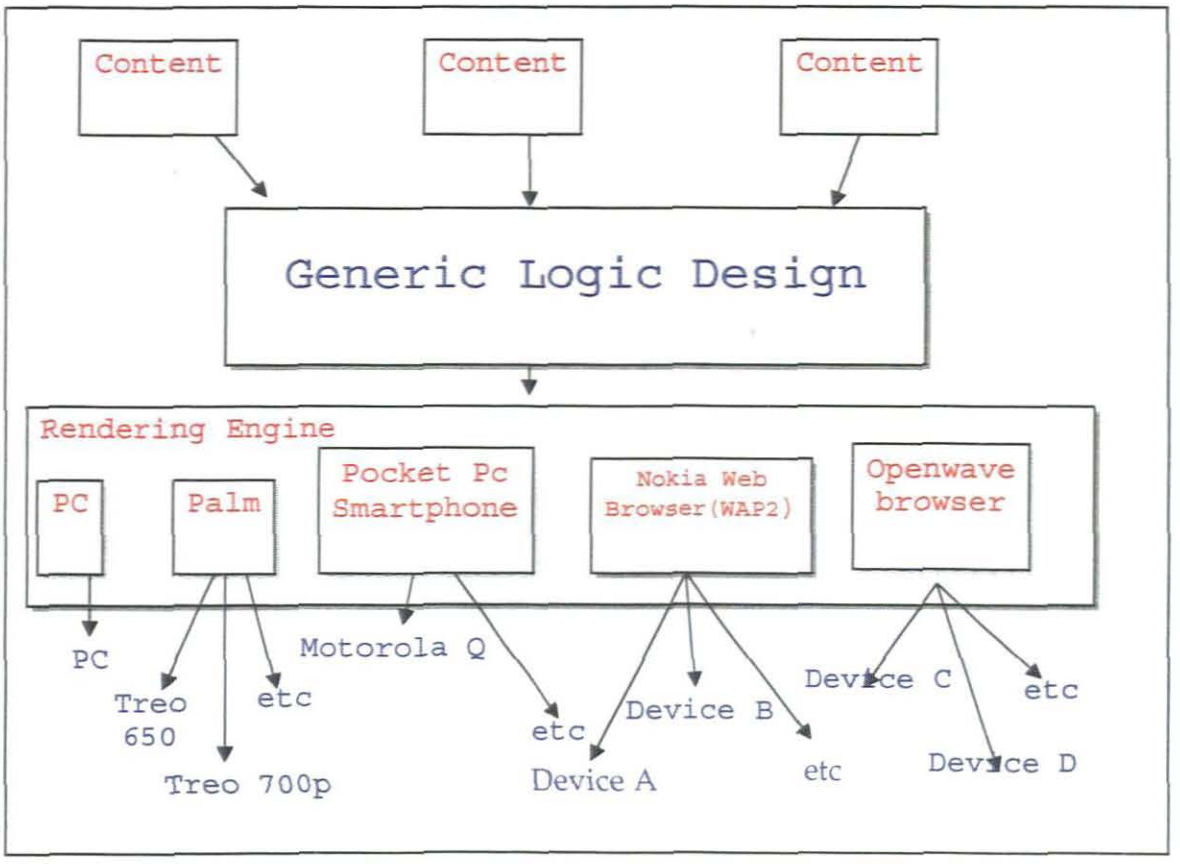
2.2 User Interfaces for Personal Communication Devices

A communication device is provided with an interface through which information is communicated to and from the outside world. In order to achieve the same interface presentation on heterogeneous devices (as in Figure 2.1), without having to separately create an interface for every device, Barbara (2007) describes an automatic translation approach.

Figure 2. 1: Three different devices with the same user interface presentation

Automatic translation involves the adaptation of a generic interface to various device types (platforms). The adaptation of this generic interface is carried out by a rendering engine. Figure 2.2 shows the automatic translation method. It shows a three tier of the automatic translation method, namely, the content, generic logic design and the rendering engine. The generic logic design of the middle tier, which is created by a designer, is adapted to different devices, such as Personal Computer (PC) and Treo 650. The Rendering Engine renders the logic design on diverse devices.

Figure 2. 2: Automatic Translation Method



In addition to the heterogeneity challenge of mobile devices, the portability of these devices, which enhances user mobility, exposes PCDs to different environmental conditions. For instance, low network bandwidth allocation or noise can adversely affect information communication. The mode of interface presentation for information access and communication on these devices must, therefore, be sensitive to changing user contexts.

Furthermore, PCDs are non-shareable and thus, every user is knitted to his or her device, thereby bringing to surface the need to give utmost consideration to user

requirements and preferences during user interface generation. In other words, user interfaces must be adaptable to user's requirements and preferences in order to achieve universal usability of interfaces. For instance, blind users may have preference for vocal interfaces. User interfaces must be designed to meet user needs. Figure 2.3 shows four different users, accessing an interface in different execution environments.

Figure2.3: Multi-Target Scenario.

A user interface must be user-requirements aware, device compatible and must be sensitive to the execution environment (David, et al., 2001).

2.3 Mobile Applications and Services Requiring User Interfaces

User interfaces for mobile applications and services, which are accessible on a PCD, can be realized using one or more of the following standards, namely, (a) eXtensible Hypertext Markup Language, (XHTML); (b) Wireless Access Protocol using Wireless Markup Language (WML); (c) Mobile Information Device Profile (MIDP) (suitable for Java 2 Platform Micro Edition (J2ME) enabled phones); (d) NET Framework and (e) Short Messaging Service (SMS) as pointed out by Nasseam, (2003). This section attempts to show how these standard technologies are used to design user interfaces for end-user accessibility. In order to achieve this aim, we selected three categories of modern systems in which end-user accessibility is paramount, namely: web applications, web services and the service grid.

2.3.1 Web applications

A web application is software stored on a server and delivered via the web, usually a combination of one or more interactive web pages that can invoke and integrate browsers, Active Server Pages (ASP), server-side Visual Basic or Java scripts, Access or SQL Server databases, HTML, XML, Cookies, and/or Component Object Model (COM) objects to create a dynamic website for bidirectional submitting, retrieving, and storing of information. A web application is usually a three-tier structure, comprising a user service tier (allowing user access to the application), a business service tier (allowing the user to carry out complex activities) and a data service tier (which allows data storage and

retrieval) (Bitpipe, 2007). Common Web applications include web mail, online retail sales, online auctions, wikis and discussion boards. A user interface for web applications can, therefore, be realized from a set of web pages designed using one or more of the above mentioned markup-languages.

2.3.2 Web Service

The World Wide Web Consortium (W3C) working group defined a web service as a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other web-related standards. The W3C working group further defined web services as Web Application Programming Interfaces (Web APIs) that can be accessed over a network, such as the Internet, and executed on a remote system hosting the requested services. These Web APIs provide interfaces for services, e.g. an interface for querying a repository for "Sport News", an interface for interacting with another Web API for retrieval of information such as "Stock Quotes". A Web service is aimed at turning the internet into a general platform for distributed computing as required for Business to Business (B2B) eCommerce.

2.3.3 Service Grid

The service grid arose in a bid to support web service technology by providing web services with some enabling functionalities. Such functionalities consist of a set of

utilities and services. Shared utilities in service grid include security, third party auditing, and performance assessment and billing and payment services. In addition to the shared utilities, are some managed services provided by service grids. These include services to aid in the transportation of messages, services for locating and obtaining the definitions and characteristics of available web services and for ensuring that web services perform as required (John and Seely, 2002). An example of a service grid is the Grid-based Utility Infrastructure for SMME-enabling Technology (GUISET) (Adigun, 2006).

2.4 User Interface Classes

User interfaces for applications and services can be categorized into three classes, namely, the multi-platform, multi-user and multi-target user interfaces (David, et al., 2001). These classes are discussed as follows:

2.4.1 Multi-Device Interfaces

A multi-device or multi-platform user interface is sensitive to multiple classes of platforms, but supports a single class of users and environments, (James, 2002; Mori et al., 2004a; Rodrigo, 2006). Multi-device interfaces are limited to devices known at design time and do not support new devices automatically. They are not context-aware. That is, they do not adapt to the user's context, which includes preferences of users and changes in user execution environment.

2.4.2 Multi- Environment User interface

A multi-environment user interface supports multiple classes of environments, but it is limited to a single class of devices. Some authors, see in (Anil, et al., 2007) regard multi-environment interface as context aware user interfaces.

2.4.3 Multi-Target User Interface

The third class of interface is the multi-target user interface, which supports multiple types of devices, users, platforms and environments. (Jank and Pospischil, 2005; Banavar, et al, 2004; Bisignano, et al., 2006).

2.5 User Interface Adaptation Approaches

Adaptation of a user interface to a device, a user or an environment is an important step towards realizing a multi-device, multi-user or a multi-target user interface. Niklfeld, et al., (2005) explains that user interface adaptation is carried out based on three common approaches, namely, the middleware adapter approach, the model-based approach and the mark-up languages base adaptation approach. These approaches have been widely used in the realization of any class of user interfaces and they are briefly discussed in the following sub-sections:

2.5.1 Middleware Based Adaptation Approach

This approach basically translates a single source interface implementation into a device specific format before the interface is finally projected to the user. The translation is handled by a middleware, which is positioned in the delivery path. Mark-up languages employed in this translation includes IBM's Abstract User Interface Markup Language (AUIML), Universal Interface Mark-up Language, (UIML), and Quality of Information Markup Languages (QIML) (Huang, 2001). These languages are collectively known as User Interface Description Language (UIDL). They run simultaneously on different devices, describing user interface in an abstract way. Using middleware based adaptation approach; interfaces are translated or adapted to a specific device. This approach is widely adopted for addressing the device diversity challenge.

2.5.2 The Model Based Approach for User Interface Design

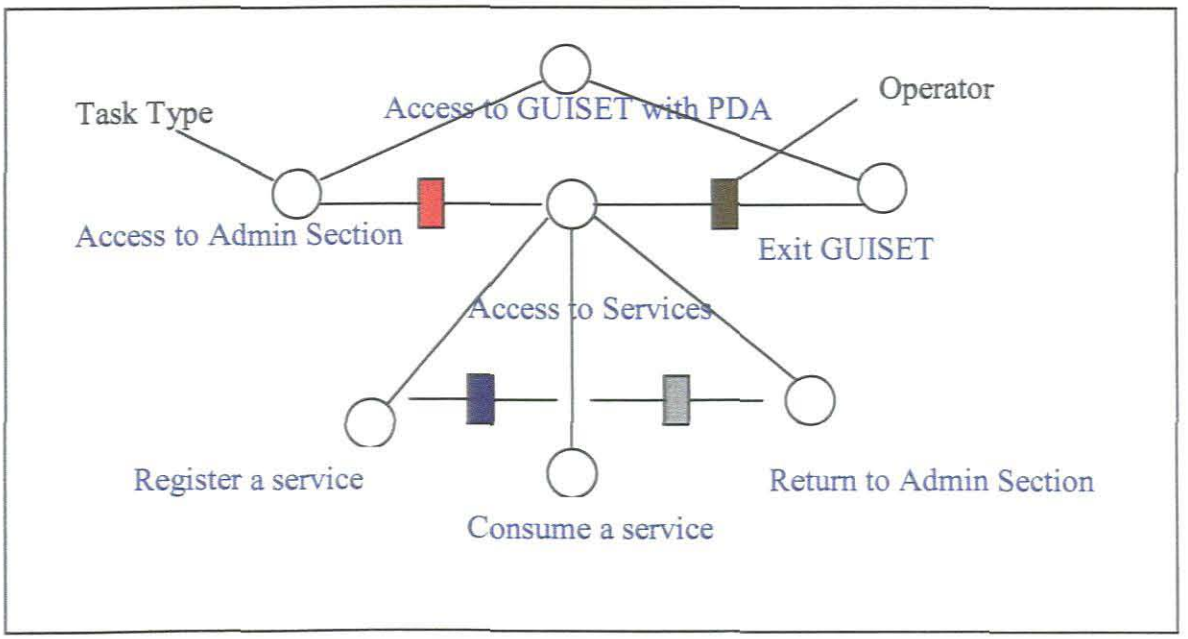
The second approach is the model based approach. Firstly, a model is, according to Szekely (1996), defined as the main component of a system which organizes information into three different levels of abstraction, namely, task model for user task descriptions, abstract user interface specification, which comprises the abstract interaction objects (AIO), information elements and presentation units and the concrete user interface object, which specifies the style for rendering the presentation units and their associated information. The concrete user interface is transformed into a final user interface using appropriate implementation language supported by a specific target device for example, eXtensible HyperText Markup Language (XHTML), Hypertext Mark-up Language

(HTML) and Wireless Mark-up language (WML). The task, the abstract and the concrete models of the model based approach are discussed in the following subsections.

2.5.2.1 Task Model

Task Model is described as the logical activity, which must be carried out in order to reach a user's goal (Paterno, 2005). The logical activity is made of tasks, tasks attributes and temporal relations among tasks. The logical activity is hierarchically represented as illustrated in Figure 2.5.

Figure 2. 4: Sample Task Model for a GUISET User Interface



Much work has been done in task model representations. Annett and Duncan (1967) proposed a hierarchical analysis of task. In (Mori, et al., 2004b), a model based tool named ConcurTaskTree (CTT) was implemented for developing and analyzing task models. The task model is transformed to an abstract model otherwise referred to as the abstract user interface.

2.5.2.2 Abstract User Interface

The Abstract User Interface describes the logical structure of an interface in a device independent way. It is a description of all interface interaction objects that support task performance. Such a description involves a number of abstract presentations, each of them identifying the set of user interface elements perceivable at a given time. The interface interaction objects are also referred to as Abstract Interaction Objects (AIO), which are identified in terms of their semantics. A user action such as "submit" is described in an abstract way without specifying the mode of representation, Such modes might include vocal, graphical or the use of gesture. Examples of interaction elements include:

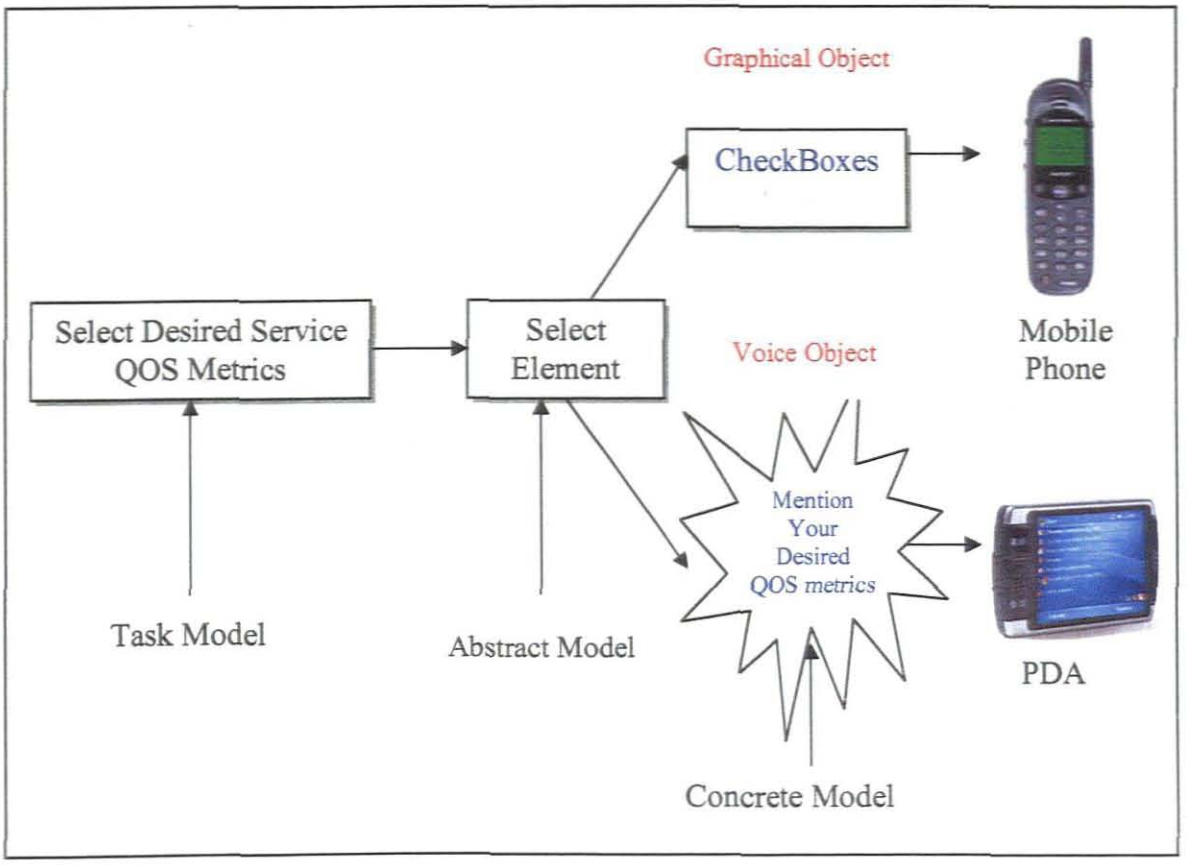
- EditInput Elements : To Edit An Object
- Selection Elements : To select between a set of elements
- Control : To trigger an event within the user interface
- Only-Output elements: Text, object description etc

2.5.2.3 Concrete Model

Cristina, et al. (2004) define a concrete user interface as the interface formed from the replacement of abstract interaction objects with the concrete interaction object, which is device or platform-specific. For example, a task like "Say Submit" can be used as vocal representation for "Submit" and its graphical representation can involve the use of a button with caption "Submit". Translation of the concrete interface into a specific

language, such as WML, XHTML, VoiceXML, Java, and HTML and so on is carried out at the final user interface level.

Figure 2. 5 : Concrete User Interface Description



In Figure 2.6, a task such as "Select Desired Service QoS Metrics" can be represented with a "Select Element" object during the abstract interface description otherwise called device independent description. The concrete description will involve a replacement of abstract interaction objects (AIO) with Concrete Interaction Objects (CIO) that are suitable for a target device or platform. The AIO is replaced with two CIO during the concrete user interface description. Graphical and voice objects are these two CIO. A graphical object such as an Check Box is selected for a mobile phone platform and for a

voice interface on PDA, a voice object is specified. Consequently, the CIO are converted to the final user interface using appropriate implementation languages. Handheld Device Mark-up Language (HDML), WML or Java can be used for the graphical object while VoiceXML can be chosen as implementation language for the voice object.

2.5.3 Mark-up Languages Adaptation

The third approach adopted in interface representation is the use of an extended mark-up language. This involves extending existing mark-up languages such as XHTML and Cascaded Style Sheet (CSS) with features enabling advanced device and modality independence. Braun, et al. (2004) extended XForms and XHTML by addition of new tags and attributes for spanning diverse user interfaces. In the work reported by (Grundy and Jin, 2002), custom java server pages (JSP) tag libraries were used.

2.6 Model Based Approaches to Multi-target User Interface Design.

The model based approach arose as a result of the quest to allow designers to specify and analyze software applications in a more semantic-oriented level, rather than starting to immediately address the implementation details (Paterno, 2005). It enables designers to concentrate on the important aspects without being distracted with the implementation details. The details of the implementation are then updated using an appropriate tool. This approach has found wide acceptability and applicability in the field of Multi-target User

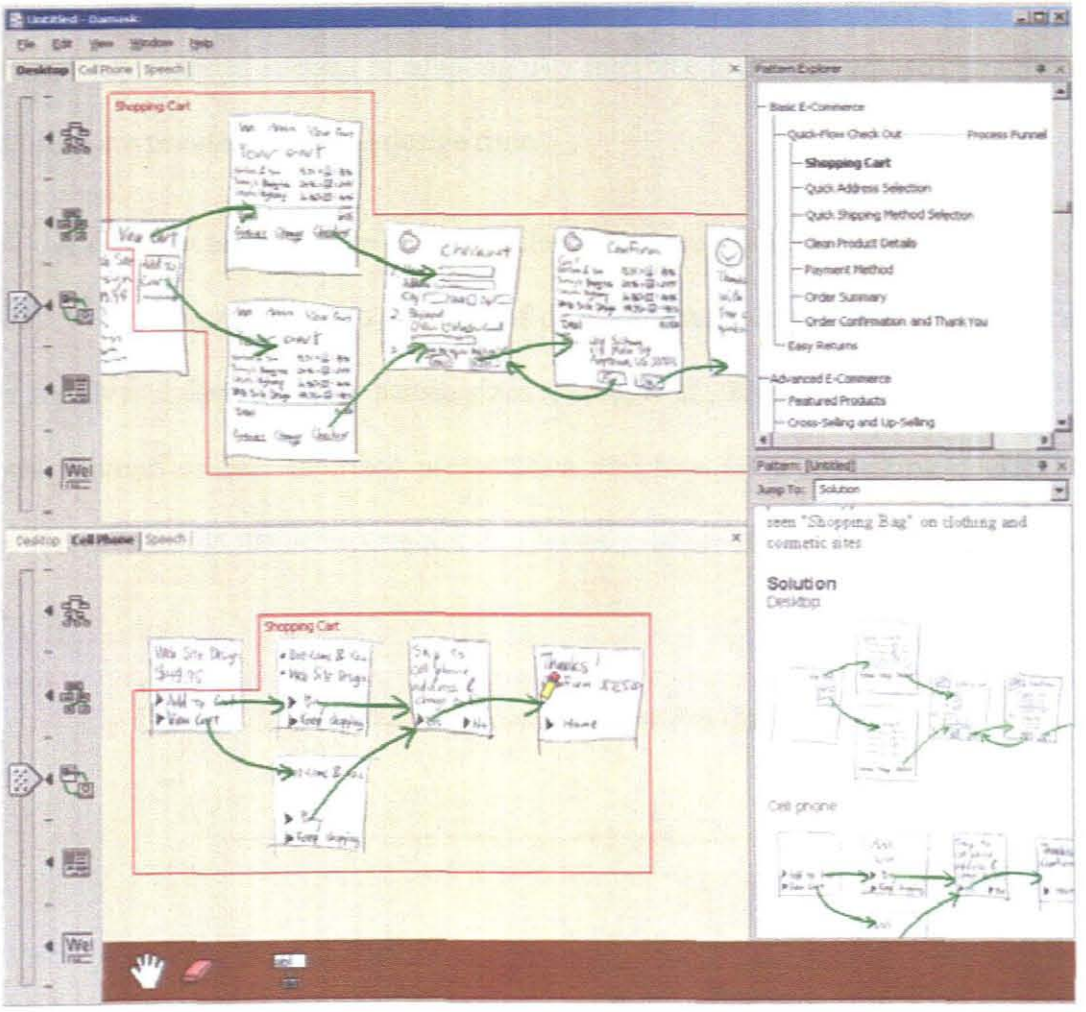
Interface design. A number of existing model based approaches for the design of user interfaces to suit the device, the user and/or the environment are presented in this section.

To achieve a multi-target interface, it is, therefore, pertinent for such interface to be device centered, user centered and/or environment centered. Many model based approaches have been proposed to address the challenges in user interface design. Some work, however, addresses the challenges posed by heterogeneity of mobile devices and some consider user requirements and needs during multi-device interface generation. In the following subsections, we categorize some exciting scholarship in multi-target user interface design based on their strength in device and user centered interface generation. A summary of our categorization is presented in Table 2.1.

2.6.1 Device Centered User Interfaces

Contributions by James, (2002) and Mori, et al. (2004a) focused on device centered interfaces. Mori, et al. (2004a) proposed the use of multiple logical descriptions for achieving device Centered interfaces and the model based approach was adopted. The approach involves task modeling, abstract and concrete user interface modeling. User interface authoring environment named TERESA was also developed. Integrated with TERESA, is a transformation algorithm, which generates interfaces for constrained (smaller) devices from the less constrained (small) devices. For instance, an interface for a mobile phone can be realized from a PDA using TERESA tool. A similar method, which focused on device centered interfaces, was proposed by James, (2002). A model based tool called DAMASK was also developed in order to aid the user interface design process.

Figure 2. 6: Snapshot of DAMASK Authoring Tool

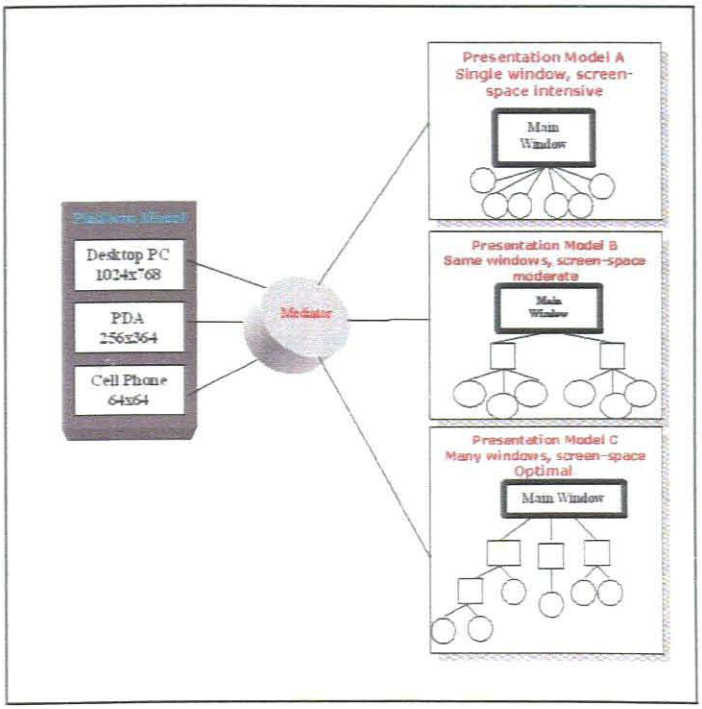


The major difference between the two approaches presented above lies in the fact that, DAMASK, which is shown in Figure 2.7 above, adopted a hybrid method of model based approach and the use of design patterns specific for each device. DAMASK allows designers to sketch out their interface during design time, and selects any design pattern suitable for a particular set of devices. The use of design patterns was adopted mainly because design patterns were considered as metaphors that are familiar to designers.

The above two approaches only cater for the design time adaptation of user interfaces, although user preference information were not considered during the transformation process. They reported success in adapting user interface to devices whose characteristics or profiles are pre-determined at design time.

A different technique for device centered interfaces was proposed by Jacob, et al. (2001). It involves design time creation of different interface presentation structures to cater for diverse device display screen sizes (Jacob, et al., 2001). At runtime, a mediator selects appropriate user interface presentation structure for a requesting device. The selection is based on the screen resolution (device profile) of a requesting device (Figure 2.8).

Figure 2. 7: Presentation Structure Selection Using a Mediator



The size of the individual interactors, (for example, the size of an edit box, the layout of interactors within a window and the allocation of interactors among several windows) are the major parameters, which determine the amount of display size required by a specific device. Adapting interfaces to a particular screen resolution of a device, such as screen resolution of 1024x768 pixels can involve an enlargement, reduction or replacement of interactors.

The method proposed by Jacob, et al., (2001) does not consider user characteristics and preference information. Also, the method is not enabled by a support tool. The method can be described to have successfully catered for devices whose characteristics, such as screen size is unknown until the time of user request.

An important contribution was made by Jank and Pospischil, et al. (2005), which led to the development of MONA Multimodal presentation server. The work provides supports for different devices and the user's choice of interaction modality. Apart from the user's preferred choice of interaction modality, other user's attributes or preferences were not considered during adaptation. The work has a unique support for a combination of vocal and graphical user interface. A similar method named "Authoring Technology for Multi-Device Web Applications" was proposed in (Banavar, et al., 2004) and a generalization editor called Multi-Device Authoring Technology (MDAT) was developed. This approach is based on the visual specification of generic applications using the Model-View Controller paradigm. Although runtime activities for handling changing user preferences or execution environment were not considered in (MDAT), the potency of the proposed approach lies in its ability to support new devices at will due to its dependency on profiles of many popular devices.

2.6.2 User Centered Interfaces

The aforementioned approaches were proposed for handling the challenge posed by the heterogeneity of mobile devices, but there are other important challenges that must be squarely addressed to achieve a multi-target user interface design. Most importantly, interfaces must be sensitive to a user needs, requirements and preferences. The following approaches consider a hybrid of user and device centered interfaces:

In the work reported by Dietrich, et al. (1993), user characteristics were considered important for the realization of user centered interfaces. User characteristics were broadly classified into two categories, namely, application independent and dependent characteristics. Application independent characteristics include user preferences, capabilities and psycho-motor-skills, while application dependent characteristics include goals and knowledge of the system and applications. Furthermore, Schlungbaum, (1997) proposed the use of an explicit user model for adapting interfaces to individual users. The work aimed at adapting interfaces to user characteristics at design time based on a defined user model. The work, however, makes the assumption that user preferences are static that is, not subject to change. This is impractical in a mobile environment, where the preferences of users changes with changes in services and user execution environment.

Savidis, et al. (1997), on the other hand, claimed that an automatic individualization and adaptation of interfaces to the requirements of individual users is very paramount to the realization of usable interfaces. A unified approach, which is based on the use of polymorphic task decomposition, was proposed. Polymorphic task

decomposition (Figure 2.9) involves the representation of tasks in arbitrary number of alternative sub-hierarchies, each sub hierarchy or style is associated with different user attributes.

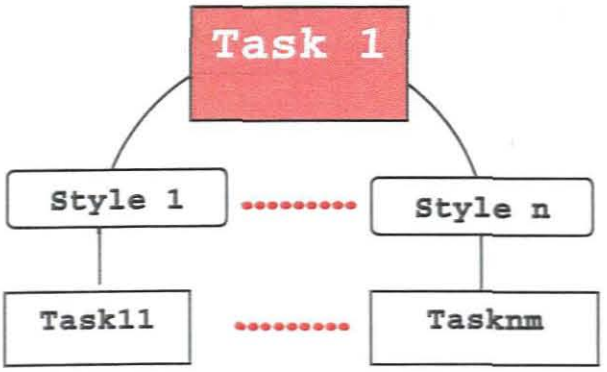


Figure 2. 8: Polymorphic Task Decomposition.
Styles Representing User attributes

User interfaces were adapted to users during design time. Additionally, an interface can adapt itself dynamically to changing user-device context information, such as user preferences and network bandwidth allocation. The unified approach is based on hierarchical task decomposition such that if a user U_1 has attribute X then a polymorphic task T with style X is activated for such user. Similarly, style Y of Polymorphic task T can be enabled for another user U_2 with attribute Y (Figure 2.10). Polymorphic task decomposition method has proved efficient for handling diversities in user attributes or requirements. It has also been adopted in this research work and reported in (Ipadeola, et al., 2008a).

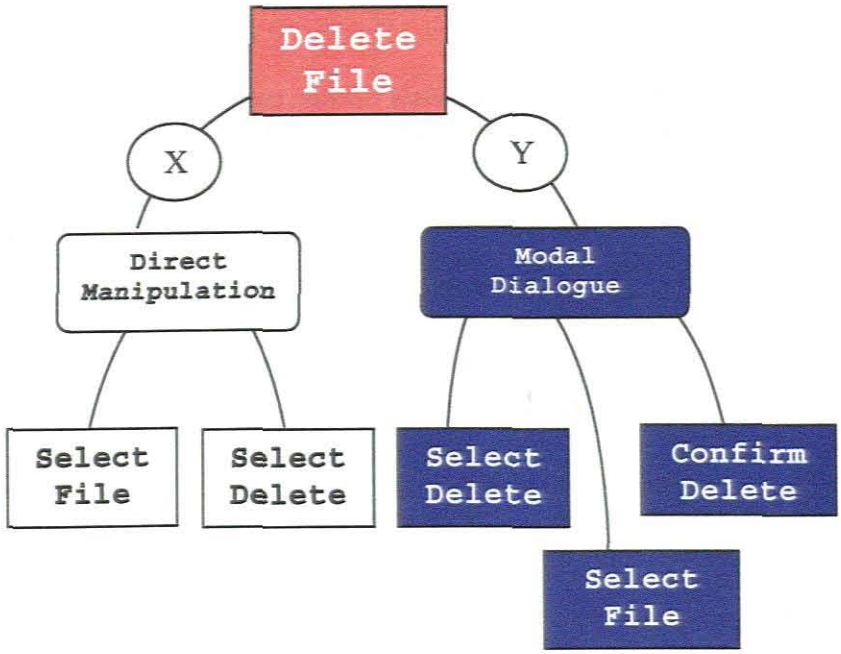


Figure 2. 9: Hierarchical Decomposition.
"Two Alternative Dialog Styles for a "Delete File"

The intent oriented approach proposed by Bisignano, et al. (2006) arose in a bid to realize user interfaces that are sensitive to user's execution environment. Flexibility was introduced as end-users were allowed to specify the use of media contents prior interface presentation.

The contributions of existing approaches in multi-target user interface design discussed above are presented in Table 2.1

Table 2. 1: Analysis of Contributions in Multi-Target User Interface Design

Contributors	Tool Support Or Tool Implementation	Platforms & Environment Supported	Strength	Limitations

James, (2002) and Mori, et al.,(2004a)	(YES) DAMASK, TERESA tools	Browser Based Environment e.g Web pages and WAP portals	Have capability for generating multi-device interfaces during design time	Limited to device whose profiles are predetermined. No consideration for user preferences or involvements Limited to browser Based Environment
Jacob, et al., (2001)	(NO)		Has capability for generating multi-device interfaces both at design time and upon request by a user.	No Consideration for user preferences or involvements.
Banavar, et al., (2004)	(YES)- MDAT	Browser Based Environment e.g Web pages and WAP portals	Supports device whose profiles are not predetermined but retrieved at runtime.	Limited to browser Based Environments
Jank and Pospischil, et al. (2005)	NO		Jank and Pospischil, (2005) allow users to specify preferred choice of interaction modality.(Graphics & Vocal interfaces)	
Schlungbaum, (1997)	(NO)	Browser Based Environment e.g Web pages and WAP portals	Adopted a user model in order to adapt interfaces to user preferences and needs	The method failed to cater for changes in user preferences and is restricted to predetermined preferences. Limited to browser Based Environments.
Savidis (1996)	(Yes)	Browser Based Environment e.g Web pages	Adopted a user model at design time and accommodates users whose preferences are not	Limited to browser

		and WAP portals	predetermined	Based Environment.
Bisignano, et al. (2006)	(NO)	Browser Based Environment, J2ME platform.	Has capability for generating multi-device interfaces both at design time and upon request by a user. Allows users to determine choice of media content. E.g the use of images.	User participation limited to choice of media content.
Ipadeola, et al., (2008)	Yes	Support J2ME platform.	Achieves Multi-device interfaces automatically. Supports device whose profiles are not predetermined. Allows direct user participation as the use of each interface artifact (including media content) is based on user preferences that are either predetermined or obtained at runtime.(During user requests)	Does not cater for Vocal interface.

2.6.3 Model Based Tools for User and Device Centered Multi-target Interfaces

Most research works on model based approaches involved the development of authoring environments, toolkits or support tools for the design or generation of user interfaces for computing devices. Some of the existing model based tools include ITS by Wiecha, et al. (1990), FUSE tool by Lonczewski, et al. (1994), Mastermind by Szekely, (1996), Mobi-D, by Angel, et al. (1997), DAMASK by James, (2002), DAMASK by James, (2002),

TERESA by Mori, et al. (2004a), CocurTaskTree (CTT) for task modeling developed by Mori, et al. (2004b). Adopting a support tool for user interface design for heterogeneous devices makes the model based approach very practical and efficient for user interface design. Model based tools shield user interface designer away from the implementation details, which are automatically generated by tool.

2.7 The Java 2 Micro Edition Platforms and the Browser Based Environments.

Although most of the approaches discussed above are directed towards browser based environments, such as Mobile Web, WAP (Wireless Access Protocol), they have immensely contributed to the success of this work, which is based on J2ME platform. The J2ME platform offers some benefits, which are unavailable in a browser based environment. The benefits of the J2ME platforms are discussed as follows (Guan, et al., 2001).

2.7.1 Advantages of J2ME Platform over the Browser Based Environment.

A. Adequate Security System

Adequate security system is very important for applications and services access. J2ME enabled phones introduce a new level of end-to-end security to the wireless world as against the WAP enabled devices, which rely on gateways to convert

between internet-oriented protocols (such as TCP/IP and SSL) and the wireless network. Such a gateway, which is stationed between a WAP mobile device and a back-end server, can be a potential security breach during protocol conversion.

B. Disconnected Access and Synchronization

Java mobile applications can run even when a device is disconnected or when a device is out of the coverage area and then synchronize with the backend infrastructure at a later time. A user will, therefore, be unaffected by incessant disconnection that can be experienced with the WEB/WAP- based interfaces.

C. Cross-Platform Compatibility

J2ME platform allows the same application or service to be run on heterogeneous devices such as cellular phone, PDA or pager. In essence, the platform provides portable interfaces that can be of huge benefits for designing user interface for mobile user applications and services.

D. Dynamic Delivery of Applications and Services

With J2ME platform, next-generation phones, PDAs, and pagers will be able to download applications safely in a real time. The platform allows a user to dynamically download new applications and services over-the-air, and then run them on any device.

E. Enhanced User Experience

J2ME platform offers richer graphics with faster interaction, user interface developers can write richer, more powerful and useful applications than the existing browser-based environments. Because graphics are generated locally, network bandwidth demand is considerably reduced.

F. Scalability and Performance

J2ME platform allows a tourist to work in standalone mode, which results in fewer users accessing the server at any given time. This improves performance and scalability for the server and reduces demand for network bandwidth.

2.7.2 Adaptable Interfaces for J2ME Enabled Devices.

Existing approaches described above are limited to browser based environment. Furthermore, user interfaces generation using the existing methods leave no room for direct user participation. Hence, the interfaces are not adaptable. An interface which allows direct user participation is referred to as adaptable interface as defined in (Savidis, et al, 1997). Direct User participation involves ability for an end user himself to change (e.g. adapt) the functionality and/or some characteristics of the user interface. User centeredness for multi-target user interfaces can therefore be best achieved when user participation is given utmost consideration.

2.8 Summary

Since the diversity challenge of computing devices has been addressed by J2ME platform, the need for an approach that achieves direct user participation and can automatically generate interfaces on-demand (runtime) to suit the preferences of a requesting user becomes very significant. Such an approach would involve a dynamic interface generation and packaging based on an on-demand user request as we have rightly presented in Ipadeola, et al. (2008b). In addition, such an approach should be able to dynamically adapt a user interface to changing user preferences.

In summary, this chapter has presented some related work (Table 2.1) in multi-target user interface design for constrained devices. The contributions from existing works are categorized according to their primary objective. Some approaches targeted addressing device limitations, while some in addition to addressing device constraints, considered user requirements. Furthermore, the need for an automatic generation of multi-target user interface for J2ME platform has also been presented.

CHAPTER THREE

MODEL DEVELOPMENT

3.1 Introduction

The previous chapter emphasized the need of user participation in multi-target user interface design and reviewed existing contributions in multi-target user interface design. In this chapter we present a model for achieving direct user participation in user interface generation. We refer to this model as polymorphic logical description (PLD) for automatic interface generation based on direct user participation. An architectural framework for integrating PLD model with users in a mobile computing environment is also presented. Furthermore, we discuss user interface modeling algorithms developed for the realization of multi-target user interfaces.

This work presents an omnibus framework that consists of a basic PLD model and mechanisms called iPLD that integrates the PLD model as a useful mobile computing tool. The framework has two distinct parts, namely, the design time tool referred to as the PLD model and load and run time mechanism known as iPLD.

The rest of this chapter is summarized as follows: Section 3.2 introduces the proposed PLD model. This Section also presents the three phases of PLD, namely, the Polymorphic Task Modeling (PTM), Polymorphic Abstract Modeling (PAM) and Polymorphic Concrete Modeling (PCM). In section 3.3, a framework that provides the infrastructures for Integrating **PLD** model into a mobile computing environment, that is,

the iPLD framework was discussed. The iPLD framework is made up of three tiers, namely design, load and runtime tiers. The PLD model makes up the design time tier of iPLD. Section 3.4 discuses the load and runtime tiers of iPLD. In Section 3.5, the information repositories of our iPLD framework were presented. The mechanisms provided by our framework for interfacing with a user for request and response communication was discussed in Section 3.6. The section also describes the User Information Bus (UIB), which aids information communication to and from the user. Section 3.7 presents the adaptation components of iPLD, namely, the Interface Adaptation Engine (IAE) and the Content Adaptation Engine (CAE). The Packager component of the iPLD framework is presented in Section 3.8. The packager is concerned with the final interface packaging and the delivery of an interface to a requesting user. In Section 3.9, the interface modeling algorithms developed in this study was discussed and lastly, a brief summary which concludes the chapter was presented in Section 3.10.

3.2 The Polymorphic Logical Description (PLD) MODEL

The polymorphic logical description (PLD) model is an interface description created at design time to address the diverse needs and preferences of users in a mobile computing environment. Conventionally, user interface design is realized from the model based approach, which comprises of three modeling phases, namely, the task model, the abstract model and the concrete model. The model based approach has been widely used for interface generation without providing support for user participation (Mori, et al., 2004b; Jacob, et al., 2001). However, the dynamic nature of a mobile computing

environment, which is characterized by changing user needs, preferences and requirements, necessitates that user interfaces are generated dynamically in response to real time user preference information rather than the use of preset interfaces. PLD claims that intrinsic characteristics of users and devices during time of request should be highly considered during interface generation.

The PLD process-model therefore, draws useful knowledge from three existing approaches that are considered significant to this study. The first of which is the polymorphic task decomposition proposed by Savidis, et al. (1997). User tasks were represented in alternative styles to cater for the diversity in user preferences and requirements. The alternative representation of task is based on an explicit user model as proposed by (Schlungbaum, 1997). Additionally, instead of the multiple logical description proposed by Mori, et al. (2004a), a polymorphic logical description is proposed in this current work. PLD involves a design time creation of polymorphic task model, polymorphic abstract model and a polymorphic concrete model. The use of semantically rich polymorphic metaphors as against the conventional models defined for the model based approach has proved significantly efficient for the realization of multi-target user interface design for end–users as reported in (Ipadeola, et al., 2008a).

Furthermore, instead of designing different interface structures for anticipated device contexts as proposed in (Jacob, et al., 2001), this current work proposes a design time creation of task variants that make up an interface, such that an interface is generated by the selection of suitable variants. The design time creation of alternative interface structures for different sets of user preferences is impractical and unrealistic for handling diversities in user preferences. PLD discretely addresses user needs and

provides automatic generation of multi-target interfaces for end–users. Hence, we present our PLD model.

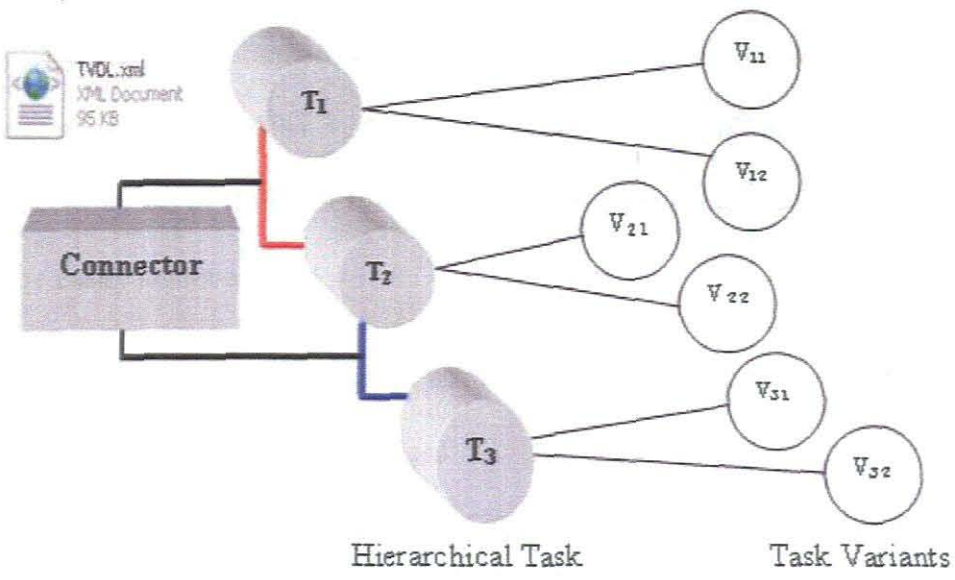
The proposed PLD process-model consists of three phases, namely the Polymorphic Task Modeling (PTM) phase, Polymorphic Abstract Modeling (PAM) phase and Polymorphic Concrete Modeling (PCM) phase.

3.2.1 Polymorphic Task Modeling (PTM)

The polymorphic task modeling is the first phase of the PLD model. The PTM is based on hierarchical task decomposition proposed by Savidis, et al. (1997). In the PTM context, we define polymorphism as the representation of an interface in multiple forms. Alternative presentations of interface have proved effective in handling the diversity in user needs and preferences as presented in (Ipadeola, et al., 2008b). An interface designer specifies different tasks that are to be performed by a user on an interface. Tasks are logical activities that must be performed by a user in order to reach a goal (Paterno, 2005) and a set of related tasks makes an interface. First, a polymorphic task model is realized at design time through analysis of tasks and their associated variants. Tasks are represented in a tree structure; therefore there is a hierarchy relationship among the tasks or nodes in the tree structure. The properties associated with a task are task types, operators and relationships between tasks. Furthermore, the tasks are categorized into the interaction task, the user task, the application task and the abstraction task. Some tasks are enabled at the same time and enabling a task can disable another. A rich task tree is constructed showing the detailed properties of tasks and their relationships.

In order to cater for diverse user preferences, tasks are given alternative representations, which are called task variants or substitutes (Figure 3.1).

Figure 3. 1: Polymorphic Task Representation



Task variants are introduced to cater for the diverse user preferences during interface generation. The diversity in user preferences can be addressed by specifying such variant information during the polymorphic task modeling. Variants can cater for different interaction styles or choice of media content, the diversity in service subscription information, diversity in language, diversity in the level of expertise of a user and the diversity in access right to sections of an interface in an organizational context. In addition, the diversity in device capability can be addressed by associating tasks with different variant representations. With the use of variants, interface designers do not have to design separate interfaces to meet user or device specific needs.

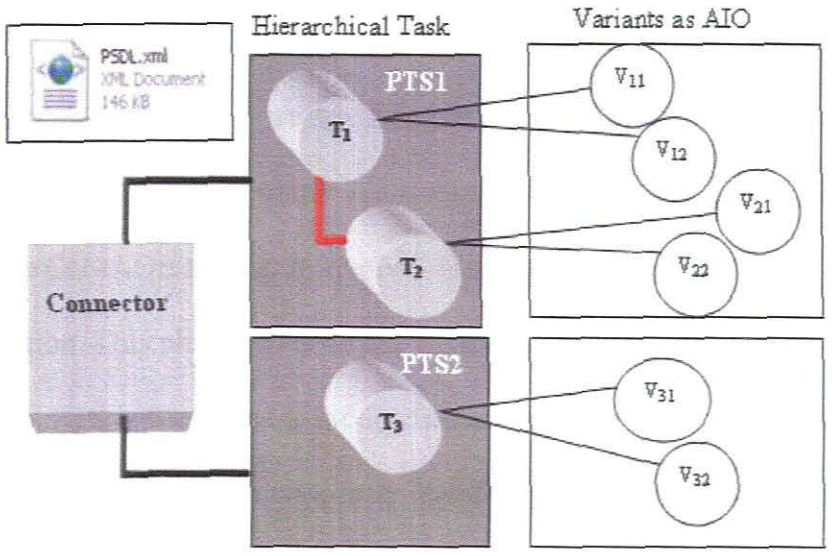
For instance, the set of tasks, namely "Enter Username" and "Enter Password" are required to be performed by a user in order to access or "sign-on" to an e-mail account. These two tasks can either be decomposed into one or more sub-tasks or given variant representations. A task can be represented in three or more different languages to achieve localized interfaces. An alternative language representation that is given to a task is an example of a task variant. Also, variants can compensate for different device capabilities. For instance, the use of different media contents for the representation of the same task can also be easily achieved.

During PTM, variants are given attribute-value description. For instance, the task variants specified in different languages like English or isiZulu is given the attribute "Language" and value "English" or "isiZulu" respectively. The variants and consequently, the attribute value description considered for an interface during PTM, depend on the preferences of the target user group obtainable from an explicit user model (Schlungbaum, 1997). Moreover, the attribute-value description associated with variants serve as the basis for the selection of a variant during interface generation. The CoMADE toolkit implements the polymorphic task modeling phase and it is used by an interface designer to specify the tasks, variants, properties and during this process task relationships are created. Tasks and variant information specified in this phase becomes translated into hierarchical structure and represented in an XML based language, called the Task Variant Description Language (TVDL). The TVDL describes each variant, its properties, associated tasks and its "attribute-value" description.

3.2.2 Polymorphic Abstract Modeling Phase

The second phase of the PLD model is the creation of polymorphic abstract model, which is also referred to as the polymorphic abstract interface. The polymorphic abstract model is realized from the transformation of tasks variants in TVDL to Abstract Interaction Objects (AIO). Abstract user interface model is a device independent description of an interface. It describes all possible user interface presentations with their associated connections.

Figure 3. 2: Polymorphic Abstract Interface Representation



In this phase, task variants are grouped into different sets, called Presentation Task Set (PTS). A navigation operator is associated with every PTS. A PTS is therefore, a window or a set of task variants perceivable by a user on his or her device at a given time. Figure 3.2 shows a sample representation of polymorphic abstract presentation. The variants (V_{11}, V_{12}, ... V_{ij}), of tasks T_1, T_2 and T_3 were represented as Abstract Interaction

Objects (AIO), which are objects with device-independent description. PTS1 and PTS2 represent Presentation Task Set I and Presentation Task Set 2 respectively.

From TVDL, (which is an XML language representation for task and variant information) different presentation task sets are constructed with their associated connections. Every presentation set contains user tasks and its associated variants. In this phase, tasks and their associated variants are represented in terms of abstract interaction objects (AIO). The resulting description of the presentation sets is represented in a Presentation Set Description Language (PSDL).

3.2.3 Polymorphic Concrete Modeling Phase

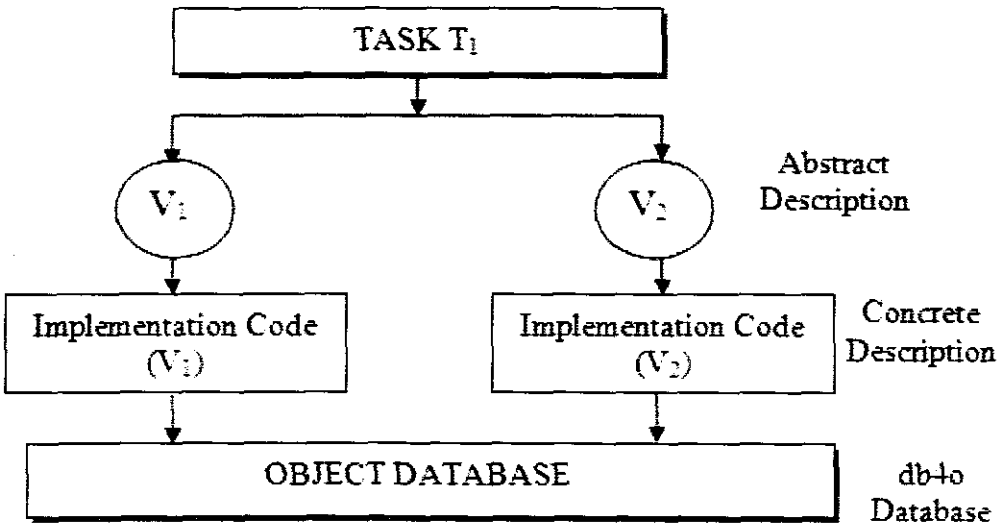
This is the third phase of the PLD model, where the generation of (polymorphic concrete model, also referred to as) polymorphic concrete interface takes place. This phase is automated by the CoMADE toolkit. It involves the generation of a device dependent user interface description. This phase relies on the information rendered in PSDL. Each PSDL form of variant in PTS is converted to a concrete interaction object, represented as $_{V}$CIO. For example, suppose a task requires a user input and if the PAM description for such task is an EditElement object, Java implementation method for a TextBox control is generated for such an EditElement object during the PCM phase. We selected J2ME as our target platform because of its unique advantages.

CoMADE therefore, generates a supporting implementation method for every $_{V}$CIO object. The method generated for every $_{V}$CIO, coupled with its description in the PSDL is persisted in an object database. Specifically, db4o was adopted for persisting the concrete description associated with each variant. For **p set** of task variants in TVDL,

there is a **q set** of AIO in PSDL and consequently a **r-set** of vCIOs objects in an object database. The concrete descriptions or the implementation methods are also referred to as application dependent method (ADM) definitions. In this phase, CoMADE also generates class templates for every user interface. Class templates include, class definition, application independent method (AIM) definitions. These aforementioned definitions are specified at design time and persisted as objects into db4o database. The designer can modify and update all generated interface component defined at this phase. Additionally, different designers can work independently on different variant implementation methods generated by CoMADE.

The proposed model also supports modularity, whereby different modules created by software experts are associated with each variant of CIOs. During this phase, CIOs, which are represented in methods with generated definitions such as return values are further enabled with functionalities. Thus, the generated methods, properties can be modified. In CoMADE, a work-board is provided to enable designers or mobile service developers to import and edit additional files. Consequently, each modification is updated. The output of the concrete interface is an object containing variants of CIO implementation methods (vCIO) and class template definitions which are persisted in the object database. Figure 3.3 shows the PSDL and CIO representation of two variants, namely, V_1, V_2 of task T_1. The CIO representation or CIO implementation methods of the variants are stored in a Db4o object database.

Figure 3. 3: Polymorphic Concrete Model Persisted Into Db4o



3.3 Architecture for Integrating PLD into a Mobile Computing Environment (iPLD)

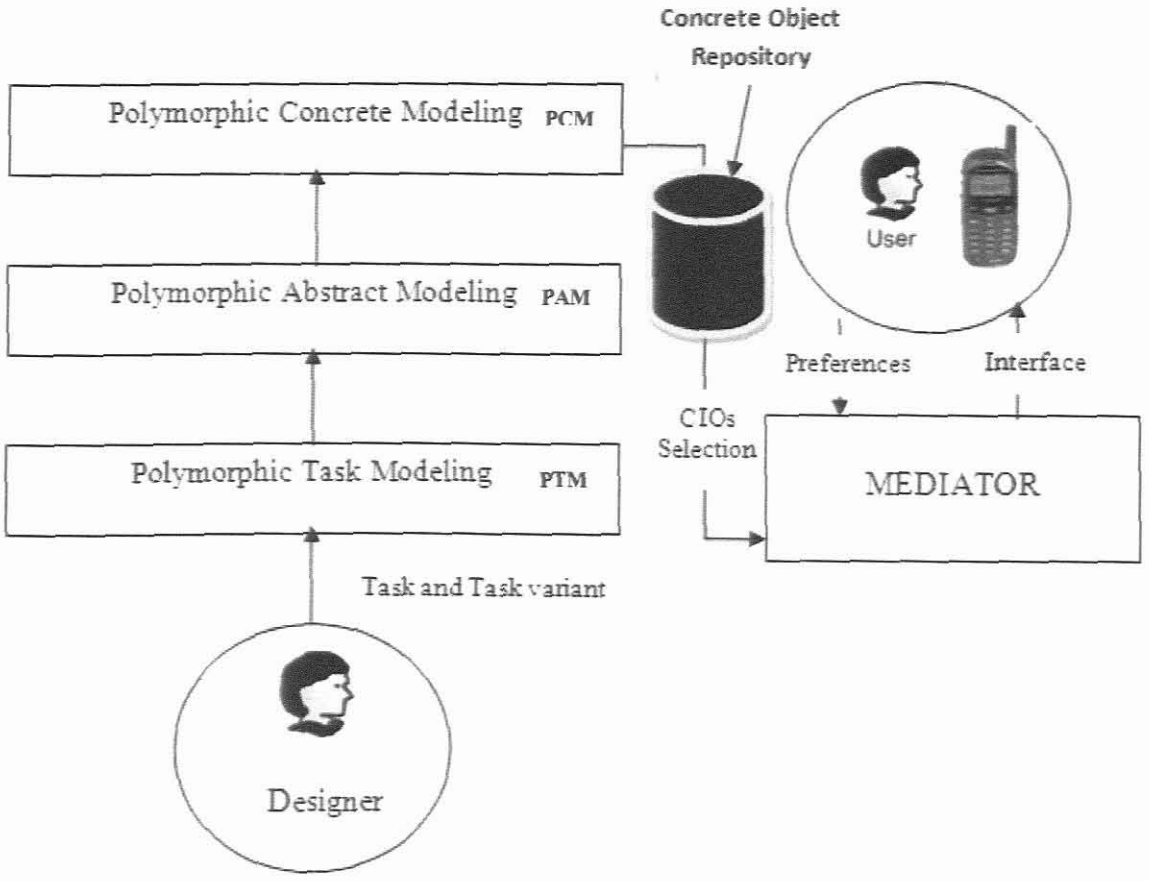
This section describes a mechanism that integrates our framework that integrates the PLD model presented in Section 3.1 with a user in a mobile computing environment. The framework is sub-divided into three tiers in order to handle the various activities of designers and end-users. Such activities include the design time creation of polymorphic artifacts by the designer and the generation of an interface based on user preference information during the load time and run time.

The three tiers of the framework, which are the design time tier, load time tier and the run time tier are discussed as follows: The design time tier is composed of the PLD model. The load and runtime tiers comprise of some components such as the Mediator, which mediates between the PLD model and a user in mobile computing environment.

Figure 3.4 represents the user centered architecture that Integrates PLD into a mobile computing environment for direct user participation during interface generation. It shows the steps involved in generation of multi-target user interface based on the proposed PLD model. Integrating PLD into a mobile computing environment requires that an appropriate mechanism be put in place to retrieve preference information from a user. The preference information forms the basis for the selection of interface artifacts from the PLD model. The interface artifacts are used for interface generation, which is then packaged and delivered to a requesting user. To accomplish the integration of PLD into a mobile computing environment, there are three important integration stages, namely, design time, load time and run time. The design time activities involve the creation of a polymorphic task model, polymorphic abstract model and polymorphic concrete model. The load time activities involve the generation and delivery of an interface to a requesting user. Load time activities are performed during the generation and delivery of a requested interface. The runtime activities include adapting an interface to changes in users' preferences. Runtime activities are those performed during the actual use of a generated interface. An example of runtime activities is response to changes in users' preferences.

Figure 3.4 shows the iPLD architecture for integrating PLD into a mobile computing environment for direct user participation during user interface generation. iPLD comprises of the PLD, a mediator component and a concrete object repository. An interface designer creates task-variant information, which CoMADE converts into a PTM and persists into TVDL.

Figure 3. 4 User Centered Process Model for Integrating PLD (iPLD) into a Mobile Computing Environment.



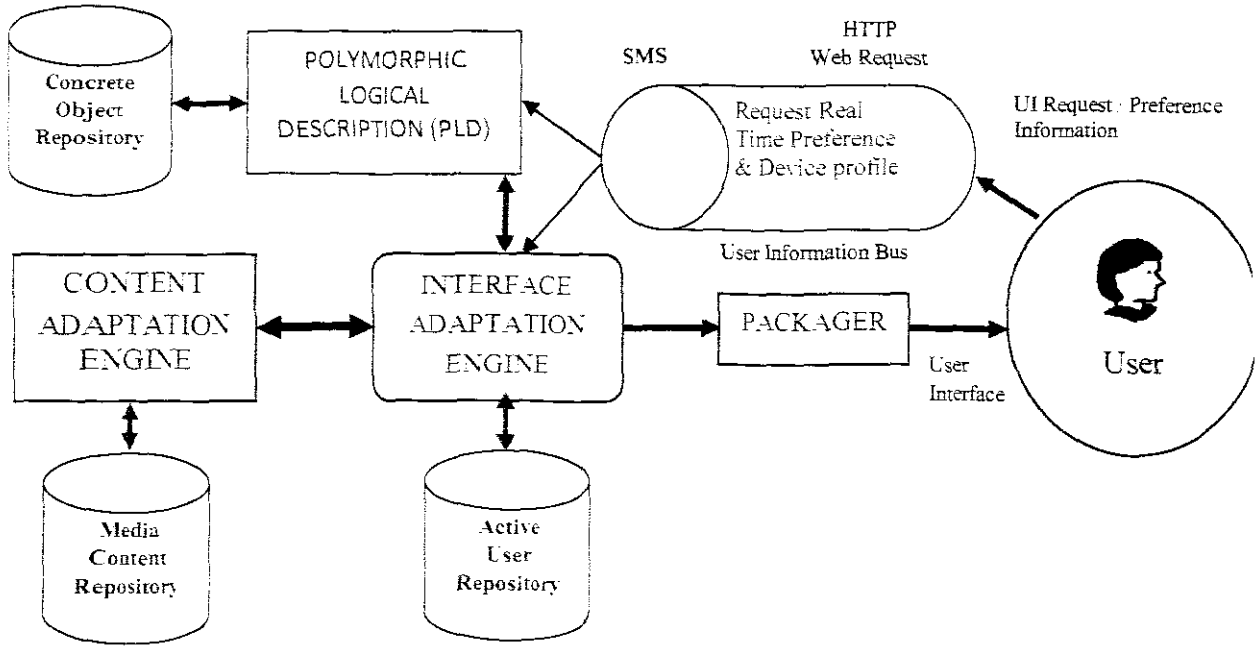
The PTM serves as input to the PAM, which is used as input to generate PCM. The preference of a user determines, which task variant of a CIO should be selected for a presentation set. For each presentation set, some task variants are selected while some are left disabled. The selected task variants make up a User-centric Presentation Task Set (UPTS). A set of UPTS is compiled, pre-verified and packaged for a requesting user. User request and preference information are conveyed from the user to the mediator for

processing. User request and preference information are received through ***web portal interfaces on proxy devices*, preference section, which is integrated into a generated interface**, and through ***Short Message Service (SMS).*** The mediator component plays an important role during the load time and runtime activities. One of the most important functions of the mediator is the selection of interface artifacts for interface generation, packaging and delivery.

3.4 Load and Run Time Tiers of iPLD

This is the load and run tiers of the iPLD architecture. Conventionally, user interface design for mobile devices was carried out separately for each device. In this work, although each user case is independently addressed, user interface generation for individual user is automatic. The multi-threading capability provided by the Java platform makes it possible for the mediator to process requests from multiple users in parallel, as different mediator instances are automatically generated in response to any request. Figure 3.5 shows the various components of the load and run time tiers. Load time and run time activities are managed by the mediator component of iPLD. This phase is handled by the mediator, which generates the final user interface for a requesting user. The final user interface is generated during the load time while interfaces can be adapted to changes in user preferences during the runtime.

Figure 3. 5 Load and Run time Tiers



The following components are associated with load and runtime tiers. They include, the mediator, which is comprised of Interface Adaptation Engine (IAE), the Content Adaptation Engine (CAE) and a Packager. Other components include the Active User Repository (AUR), Concrete Object Repository (COR), Media Content Repository (MCR) and a User Information Bus (UIB).

3.5 The iPLD Repositories

This section discusses the three major repositories of the iPLD architecture. The repositories include the Concrete Object Repository (COR), the Active User Repository (AUR) and the Media Content Repository (MCR).

3.5.1 Concrete Object Repository

This component is concerned with the storage of every concrete interaction object. In other words, every variant, with a concrete description (vCIO) from the PLD model created at design time is stored in COR. COR comprises of presentation task sets (set of vCIO objects). Additionally, the description associated with every vCIO is also stored in the COR. The COR is queried for the selection of suitable vCIO during a request for an interface generation. User preferences are used as query parameters during the selection process. User information or requests are received through an information bus.

3.5.2 Active User Repository

The Active User Repository (AUR) keeps the record of all user requests and their corresponding responses in terms of their local addresses of the generated interfaces. For every request received by the mediator, AUR is queried against such request to detect any match with existing requests. When a match is found, then an already generated interface is retrieved and delivered to a requesting user.

3.5.3 Media Content Repository

The Media Content Repository (MCR) component stores the various contents for user interface presentations. It can be referred to as an interface resource directory. Media contents, such as images, video and audio file are managed by this component. During an interface generation for a requesting user, an appropriate content is selected and adapted by the content adaptation engine. Adaptation of media contents includes sampling of

images and substitution of media contents. An audio content can be selected as substitute for a video content.

3.6 User Request/Response Information Communication

Information communication to and from the user during or prior the generation of an interface is achieved through different methods. The communication is aided by a User Information Bus (UIB). This section describes the UIB and the three methods used by iPLD for user request/response communication.

3.6.1 User Information Bus (UIB)

This component conveys information from the user to the mediator for processing. User requests or information is received by the mediator through web portal interfaces, user application and Short Message Service (SMS). All information or requests received by the UIB are logged in the AUR.

3.6.2 An Interface Shopping Cart for Preference Handling

The Interface shopping cart method allows for user preference management. It enables a user to manage his or her preference information. Additionally, user can view existing interfaces and generate sample interfaces which can be requested for delivery or added to a cart. We consider interface as a product, which has properties and other substitutes (variants). A three-channel method of our interface shopping cart is presented as follows.

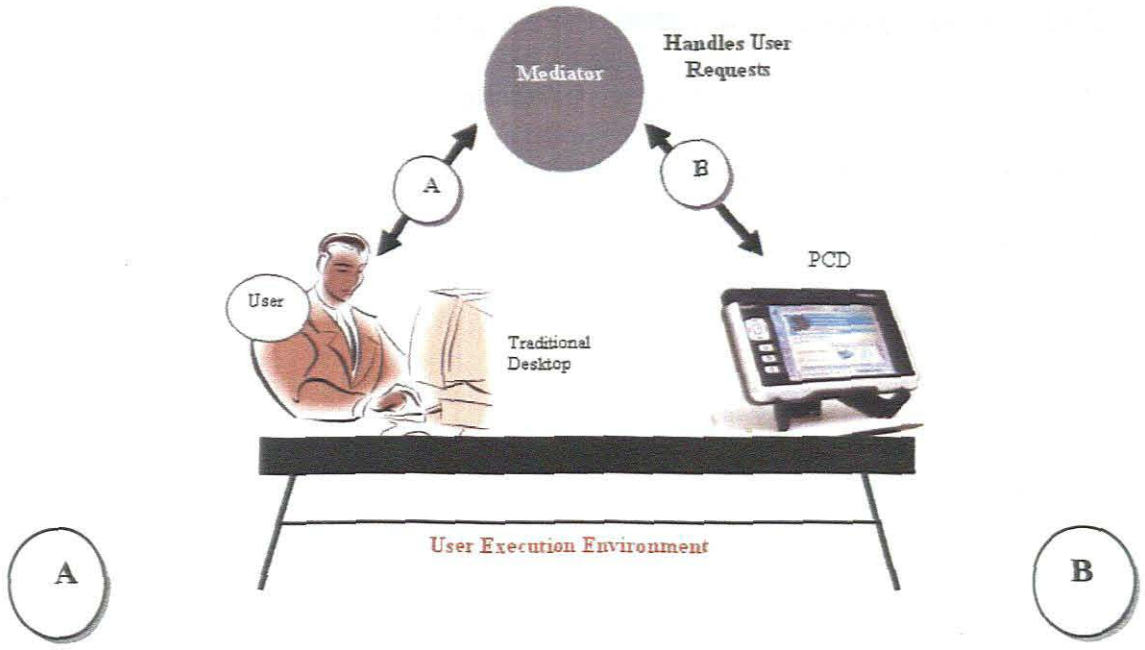
A. Proxy Device: An explicit Proxy Application on a PC.

Users are provided with a proxy application accessible on large screen devices, such as Personal Computer (PC) for specification of test or actual preferences (Figure 3.6). Users can use the proxy application for submission of their preferences. User preference information can contain the user's choice of language, service selections, media content and desired look and feel. Other examples can include some interface differentiation information, such as the desired service level, which determines the limit of the functionalities expected of an interface. Whenever a request for an interface is received, the mediator component uses the preference information for the generation of an interface. Snapshot of any generated interface is presented on the proxy application prior a request for delivery to a user mobile device.

B. Direct Access through User Mobile Devices

Users can alter their preferences through their mobile devices. This is achieved through an integration of a Preference Modification Section (PMS) with an initially generated user interface. The PMS provides users with information regarding the existing user interface presentation alternatives supported by CoMADE. It also allows users to provide or alter their preference information.

Figure 3. 6: Real-Time Preference Submission



A user can

- *View already Generated Interfaces(UI)*
- *Request For an Existing UI or*
- *Submit Preference information on traditional desktop, which is conveyed by the User Information Bus (UIB) component to the mediator*
- *Generate User centered Interfaces*
- *Add Generated Interface To Cart*
- *Check out or*
- *Make a Request for Delivery of Generated Interface to user device*

A user

- *Performs 'A' using his/her Personal Communication Device. Or*
- *Requests for an already generated interface (Included in the Cart or already generated by other Users) using SMS via a PCD.*
- *Identifies interface with unique identification code.*

C. SMS Technology for Predefined Preferences

Users can decide to update their preferences or request for a particular interface choice. As each generated interface is associated with an identification code, therefore a user can

make a request using Short Messaging Service (SMS) to the mediator for an interface presentation that has already been generated on a proxy application. Such interface presentation is identified by a unique identification code. After this, a mediator delivers an address to the location of the generated user interface for downloading by a requesting user.

3.7 Core Adaptation Components

The information retrieved from user through the UIB serves as the basic input to the iPLD adaptation components. Basically, the iPLD is made up of two adaptation components, namely the Interface Adaptation Engine (IAE) and the Content Adaptation Engine (CAE). These components play significant roles in adapting an interface (set of PTS) to a requesting user or simply put as the realization of a set of User Presentation Task Set (UPTS).

3.7.1 Interface Adaptation Engine

The Interface Adaptation Engine carries out the generation of the final user centered interface. It is the most significant component of the mediator. It utilizes the information from other components during interface generation. When user preference information is received through an explicit proxy or user device, or specified as sample preference by the designer, the preference information is conveyed to the AIE by through the User Information Bus (UIB). The AIE performs a look-up for the appropriate database object

for the requested application interface. When such an object is found, the mediator queries the object database for the selection of variants (vCIO) with a preference match with the submitted preference. At most, a variant is selected for every task object. Each selected variant object is referred to as a stub for retrieving its associated implementation method. Likewise, the class template information associated with a requested service is also retrieved. In addition to these interface artifacts, declaration definition is derived from every selected method. The order of representation is realized during this stage. These components make up the whole interface, which is compiled, pre-verified and packaged for a requesting user.

Figure 3.7 shows two different Tasks, namely, Task 1 and Task 2, denoted by T_1 and T_2 respectively. Task 1 (T_1) has the set (v_{11}, v_{12} and v_{13}) as its variant representations and Task (T_2) comprises of variants (v_{21}, v_{22} and v_{23}). A user centered presentation task set (UPTS) for a user is composed by the selection of variants based on user preferences.

In summary, the argument is that, interfaces are built up for a user by the selection of suitable polymorphic artifacts (variants) based on user preference information. The selection of suitable variants is carried out during dynamic interface generation (Figure 3.7). User preference information such as choice of service selection, language, choice of media content and mode of service delivery can be provided through an explicit web page.

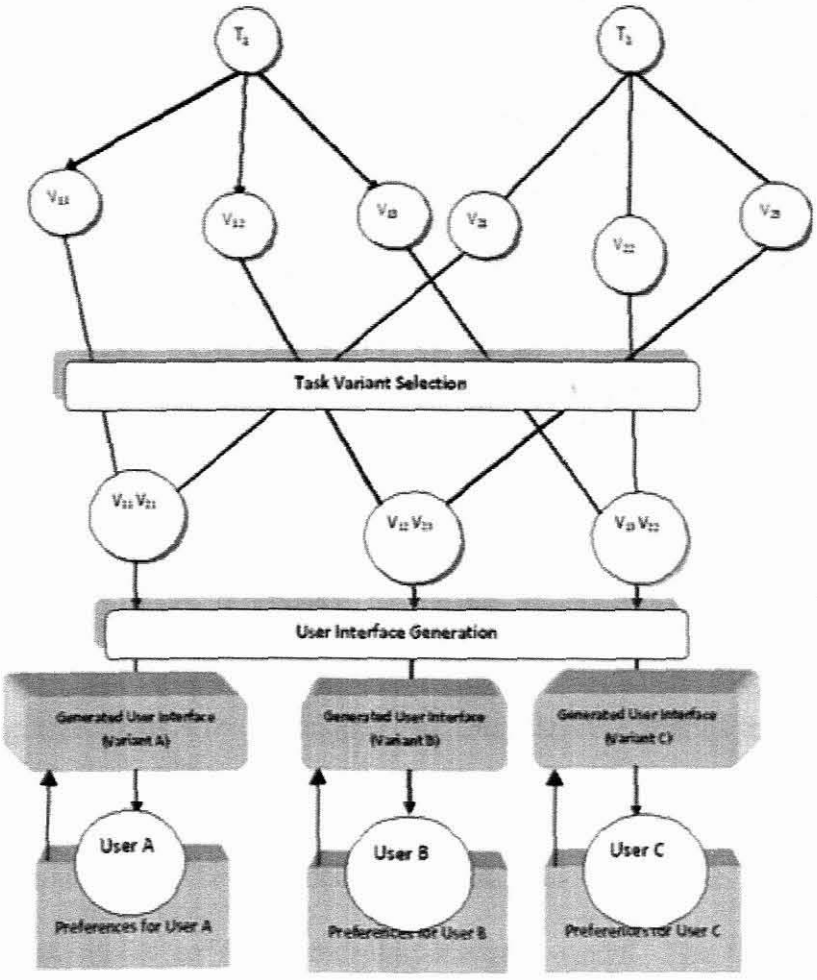


Figure 3. 7: Interface Composition Process

3.7.2 Content Adaptation Engine (CAE)

The CAE is a component of iPLD that handles media content adaptation. It relies on user preference information and the profile of a requesting device for the selection and adaptation of a media file. Images are grayed, scaled or sampled. An audio content can be substituted for a video content. Also a reduced quality of video contents can also be projected to users based on retrieved preferences.

3.8 Packager

The Packager performs final processing such as compilation of all needed files, pre-verification of class files, creation of manifest, packaging into Java Archive file (JAR) and the creation of Java Application Descriptor file (JAD) and finally the generation of a universal resource location (URL) that points to the generated multi-target user interface. The URI of the generated interface and associated preferences are dynamically stored in the AUR.

3.9 Modeling Algorithms for the Proposed Method

Four modeling algorithms were developed for achieving the goal of this study. The first algorithm was developed for creating a polymorphic task model through task and variant information specification. The second algorithm was developed for transforming a polymorphic task model to a polymorphic abstract model. The third algorithm was developed for transforming a polymorphic abstract model to a polymorphic concrete model. The fourth modeling algorithm was developed for transforming a polymorphic concrete model to final multi-target user interface based on specified preference information for concrete variant (vCIO) selection.

3.9.1 Algorithm for creating Polymorphic Task Model using the CoMADE toolkit

The following algorithm is developed for generating a polymorphic task model during the design time. A designer uses the requirement document of an application interface to be generated for the specification of tasks during this phase.

INPUT: Requirement Document describing the tasks that make up a user interface

Step 1: Designer specifies a project name. // Project name becomes the root task.

Step 2: Repeat #2 (While More Tasks in the Interface Requirement Document)

Step 2.1: Designer specifies a task name

Step 2.2: Designer specifies task properties

Step 2.3: Designer saves current Task

Step 2.4: Task tree is updated with task information

Step2.5: Repeat # 2.5 (While More Variants To Be Associated with current Task)

Step 2.5.1: Designer specifies the variant name

Step 2.5.2: Designer specifies the variant properties

Step 2.5.3: Designer Saves Variant with Current Task

Step 2.5.4: Variant tree is updated with Task and Variant information

End Repeat # 2.5

End Repeat #2

Step 3: Designer saves task and variant information

Step 4: An array of task objects with their associated variants, including variant description is created and represented in an XML based language called Task Variant Description Language (TVDL).

OUTPUT: An array of task objects with their associated variants

3.9.2 Algorithm for transforming Polymorphic Task Model to Polymorphic Abstract Model.

Suppose Task represents an array of task objects and every each task T_o object in Task has some variants V_o that make up an interface. C_t represents the size of the task objects specified for an interface. Let T_{AIO} be the AIO representation for each of task object T_o and let V_{AIO} be the AIO representation of the variants associated with each of task object T_o. Suppose the size of variants associated with a task object is C_v. A main interface presentation object is represented as MP and P represents other interface presentations. Let Task Operator Property Tt be set to "Enable";

INPUT: An array of task objects and their associated variants specified by a designer using CoMADE toolkit

Step 1: **Repeat #1** (While there are more tasks in Task)

Step 1.1: Retrieve each task from the input Array Task Object. Set the task to Current task (To).

Step 1.2: Retrieve Tt as the task operator property of current task To.

Step 1.3: Retrieve Cv, which is the total number of variants associated with To.

Step 1.4: Create a new T_{AIO} object as parent node for variants associated with To.

Step 1.5: **Repeat # 1.5** (While there are more variants associated with the current task To)

Step 1.5.1: Retrieve each variant V_o (current variant) from the current task T_o

Step 1.5.2: Convert V_o to its Abstract Interaction Object (AIO) form and name it V_{AIO}

Step 1.5.3: Add V_{AIO} as child node to T_{AIO}

End Repeat #1.5

Step 1.6: Add T_{AIO} as child object to Presentation P

Step 1.7: If Task Operator tt is Disable "if (Tt.equals(Disable") == true)

Step 1.7.1: Add P as a child Presentation to Main Presentation MP.

Step 1.7.2: Reinitialize Presentation P.

End Repeat #1

Output: Main Presentation (MP) containing Presentations of Abstract Interaction Objects (AIO)

3.9.3 Algorithm for transforming Polymorphic Abstract Model to Polymorphic Concrete Model.

Suppose MP represents an array of presentation P containing tasks T_{AIO} in AIO form and every task T_{AIO} object in P has some variants V_{AIO} in AIO form that make up an interface. C_p represents the total count of presentation in MP. C_t represents the size of the task objects specified for each presentation. Let T_{CIO} be the CIO representation for each of task object T_{AIO} and let V_{CIO} be the CIO representation of the variant objects associated with each of task object T_{AIO}. Suppose the size of variants associated with a task object is C_v

INPUT: A main presentation objects (MP) containing presentations, task and variants in AIO representation.

Step 1: **Repeat #1** (While there are more presentation P in MP)

Step 1.1: Retrieve each presentation P from MP. P is the current presentation

Step1.2: Retrieve C_t, which is the total number of Presentation P in MP.

Step 1.3: **Repeat # 1.3** (While there are more tasks T_{AIO} in presentation P (T_o in AIO form))

Step 1.3.1: Retrieve each Task T_{AIO} (current task) from the current Presentation P.

Step 1.3.2: Retrieve Cv, which is the total number of variants associated with T_{AIO}.

Step 1.3.3: Get the CIO representation of T_{AIO} as T_{CIO}

Step 1.3.4: **Repeat # 1.3.4** (While there are more Variants V_{AIO} in T_{AIO} (V_o in AIO form) in a Presentation.

Step 1.3.4.1: Retrieve each variant V_{AIO} (current variant) from the current task T_{AIO}

Step 1.3.4.2: Convert V_{AIO} to its Concrete Interaction Object (CIO) form and name it V_{CIO}

Step 1.3.4.3: Add V_{CIO} as child node to T_{CIO}

End Repeat # 1.3.4

Step1. 3.5: Add T_{CIO} as child object to Presentation P

End Repeat # 1.3

Step 1.4: Add P as child object to Main Presentation MP

End Repeat #1

OUTPUT: Main Presentation (MP) containing Presentations of Abstract Interaction Objects (AIO)

3.9.4 Algorithm for transforming Polymorphic Concrete Model to Final User Interfaces.

Suppose MP represents an array of presentation P containing tasks T_{CIO} in CIO form and every task T_{CIO} object in P has some variants V_{CIO} in CIO form that make up an interface. C_p represents the total count of presentation in mp. C_t represents the size of the task objects specified for each presentation. Let T_{CIO} be the CIO representation for each of task object and let V_{CIO} be the CIO representation of the variant objects associated with each of task object T_{CIO}. Suppose the size of variants associated with a task object is C_v. A variable U is an object containing user preference information.

INPUT: A main presentation objects containing presentations, task and variants in CIO representations.

Note: Different Designers can update different CIO variant information and also create modules needed for the business logic at design time. Every needed module is specified and associated with the project file.

Step 1: **Repeat #1** (While there are more presentation P in MP)

Step 1.1: Retrieve each presentation P from MP. P is the current presentation

Step 1.2: Retrieve C_t, which is the total number of Presentation P in MP.

Step 1.3: **Repeat # 1.3** (While there are more tasks T_{CIO} in presentation P (T_o in CIO form))

Step 1.3.1: Retrieve each Task T_{CIO} (current task) from the current Presentation P

Step 1.3.2: Retrieve Cv, which is the total number of variants associated with T_{CIO}.

Step 1.3.3: **Repeat # 1.3.3** (While there are more Variants V_{CIO} in T_{CIO} (V_o in CIO form) in a Presentation.

Step 1.3.3.1: Retrieve each variant V_{CIO} (current variant) from T_{CIO}

Step 1.3.3.2: Retrieve V_{CIO} Description (VD) property from V_{CIO}

Step 1.3.3.3: **Repeat # 1.3.3.3** (While there are more user preference objects u in U)

Step 1.3.3.3.1: Retrieve each user preference object u from U

Step 1.3.3.3.2: if the Description of a user preference object (u) MATCHES V_{CIO} Description (VD)

Step 1.3.3.3.2.1: Add V_{CIO} to userPresentation

Step 1.3.3.3.2.2: Break out of the loop **#1.3.3.3**

End Repeat #1.3.3.3

End Repeat #1.3.3

End Repeat # 1.3

Step 1.4: Add other modules specified by designer to userPresentation

Step 1.5: Add userPresentation to finalPresentation object

Steo 1.6: Re-initialize userPresentation object

End Repeat #1

Step 2: Write finalPresentation into Main File

Step 3: Compile, Pre-verify and Package MainFile

Step 4: Deliver Application Interface.

Output : MainFile ==> Java Application Interface

3.10 Summary

To achieve a multi-target user interface design, this work proposes a polymorphic logical description for handling the diversities in user preferences based on an explicit user model, user groups or roles. We have also presented our framework, which provides infrastructures for integrating PLD into a mobile computing environment. User preference information is obtained through the proposed user interface shopping carts. A dynamic run time interface generation method was realized through the iPLD components. The interface modeling algorithms are also presented in the chapter.

To this end, we have achieved our first, second and third objectives by evolving a polymorphic logical description model, development of modeling algorithms for multi-target user interface and development of a framework for integrating PLD into a mobile computing environment. The next chapter describes the prototyped design of iPLD and its implementation and evaluation details.

CHAPTER FOUR

PROTOTYPE DESIGN, IMPLEMENTATION AND EVALUATION

4.1 Introduction

The previous chapter described our proposed iPLD framework, which comprises the design, load and runtime tiers. The PLD model, which constitutes the design time tier, was described in addition to the components of other tiers. This chapter presents a prototype design, implementation and evaluation of the proposed toolkit, named CoMADE based on the proposed multi-tier iPLD framework. The toolkit caters for creation of PLD during design time and handles the activities of the load and runtime tiers. The toolkit achieves direct user participation in multi-target user interface design. Furthermore, the chapter discusses the evaluation procedure and results of the proposed toolkit. Section 4.2 gives an overview of CoMADE toolkit together with its functional description. A description of the implementation details of the Toolkit is presented in Section 4.3. We describe the CoMADE environment in Section 4.4 and Section 4.5 presents the evaluation of the toolkit. Finally, Section 4.6 gives a brief conclusion of the chapter.

4.2 Prototype Design

In this section, we present a formal description of the proposed system. The components of PLD and the sequence of steps for the realization of multi-target user interface are represented using a Unified Modeling Language.

4.2.1 An Overview of the toolkit

This section gives an overview of the proposed system, showing the major activities of the actors of the proposed toolkit. Basically, two major actors, namely, an interface designer and an end-user can directly interact with the proposed toolkit. The activities of the actors are described as follows.

A. Interface Designer

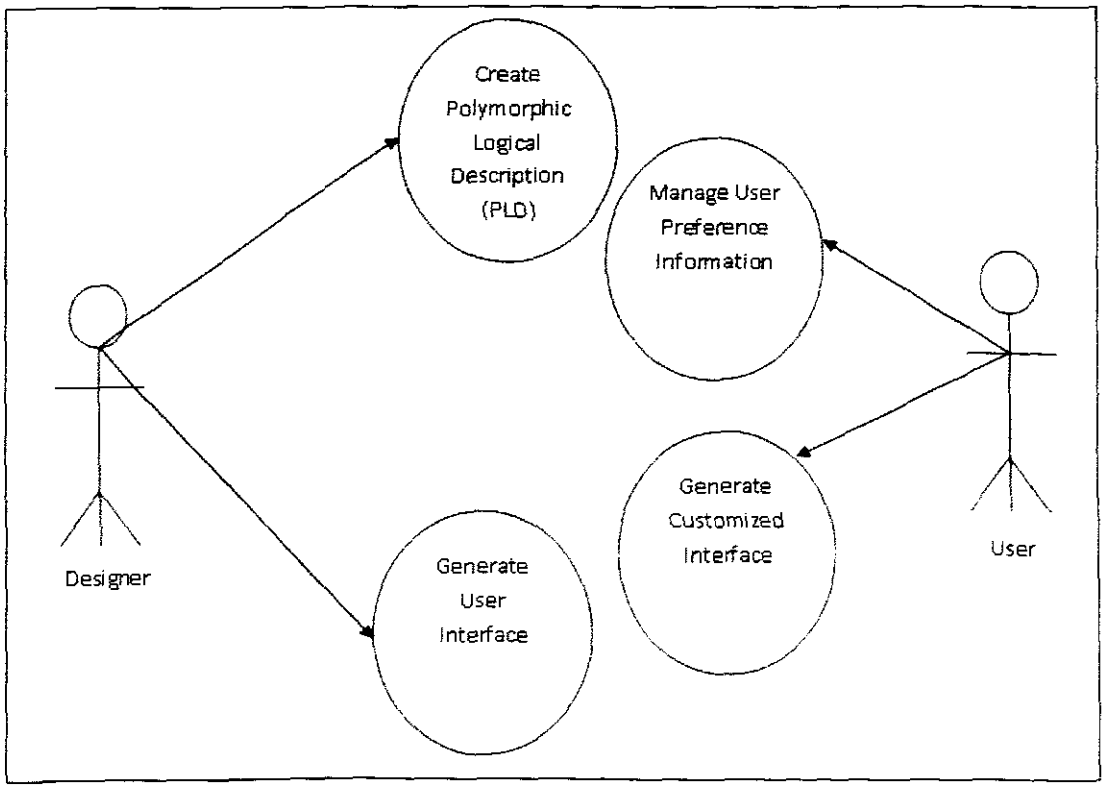
An interface designer handles the creation of a polymorphic logical description for an application during the design time. The designer creates a polymorphic task model, polymorphic abstract model and polymorphic concrete model. The designer can also generate the final user interface using example-preference information.

B. An end-user

An end-user provides his or her preference information for interface generation. The information is used in the selection of appropriate interface artifacts (Task Variants in CIO form) for the final user interface generation. Users can also alter their preferences at will using the methods proposed in the previous chapter. The actors of the system perform one or more roles during the design, load and runtime phases. Basically the

designer is mostly concerned with the design time activities and the users play significant roles during the load and runtime activities. Use case diagram for the proposed toolkit is shown in Figure 4.1.

Figure 4. 1 Use Case of the Proposed System

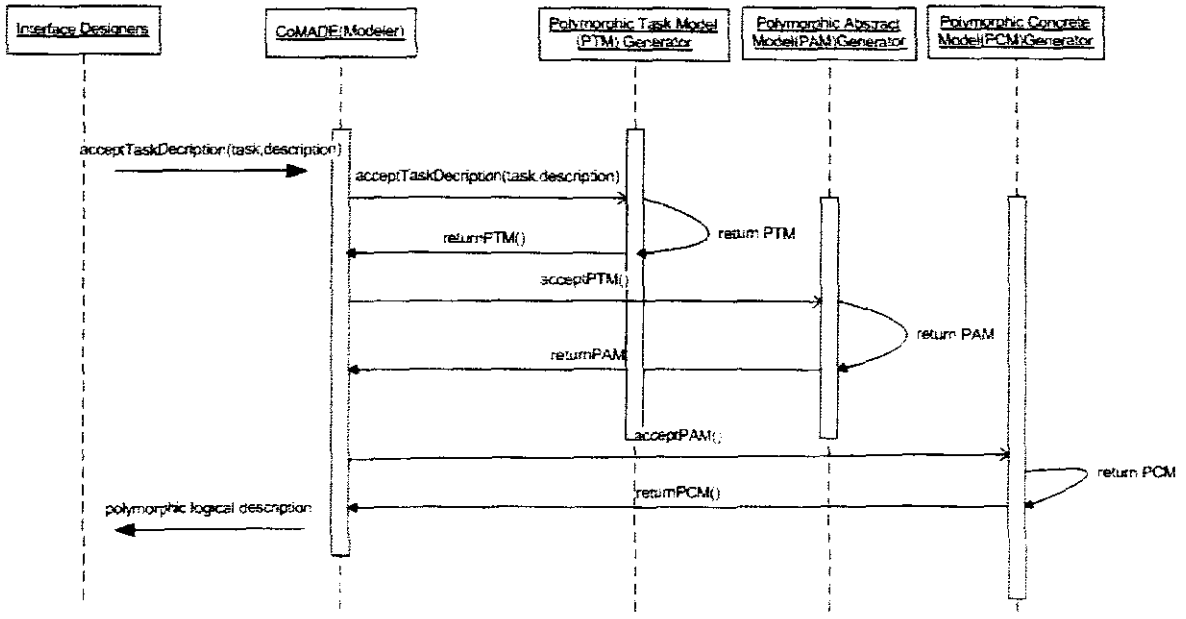


4.2.2 Design Time Activities

During the design time (Figure 4.2), the task descriptions and their associated variants are specified for the generation of a polymorphic logical description. CoMADE activates the PTMGenerator for the creation of polymorphic task model. The output of the PTMGenerator is used by the Polymorphic Abstract Model Generator (PAMGenerator) for an automatic generation of polymorphic abstract model. Finally, polymorphic concrete model is realized by the PCMGenerator using the output from the

PAMGeneratorator. The visual representation of the flow of the design time activities for creation of polymorphic logical description is shown in Figure 4.2.

Figure 4. 2 Sequence Diagram for the Creation of a PLD by an interface Designer

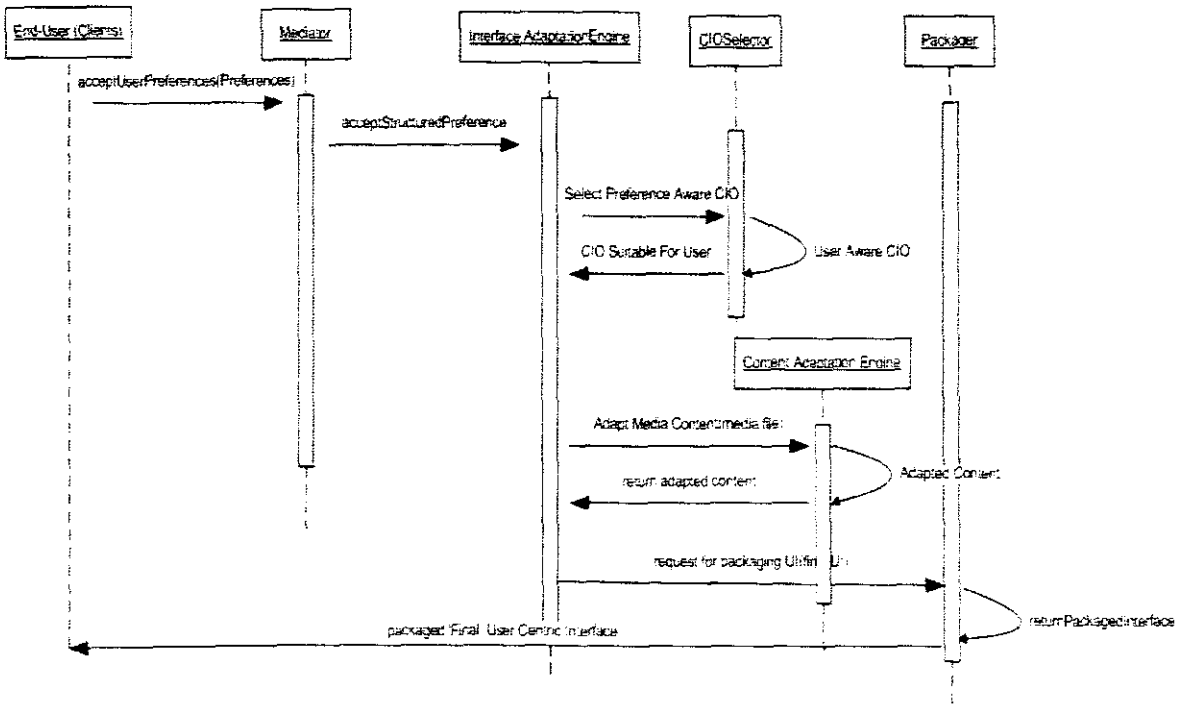


4.1.3 The Load and Runtime activities

Multi-target user interface are generated for user during the load and run time. The runtime activities are targeted towards user interface adaptation in response to changes in user preferences. During the load time, the Interface Adaptation Engine (IAE) performs the selection of polymorphic artifacts (CIO variant description) based on user preference information. IAE generates a user centric interface based on the selected CIO artifacts. Media content selection and adaptation requests are forwarded to the Content Adaptation Engine. The IAE generator sends the generated interface to the Packager for compilation,

pre-verification, creation of resource files and archiving of interface and its associated files before the final delivery to a requesting user.

Figure 4. 3 Sequence Diagram for the Generation of Customized Interface by a User



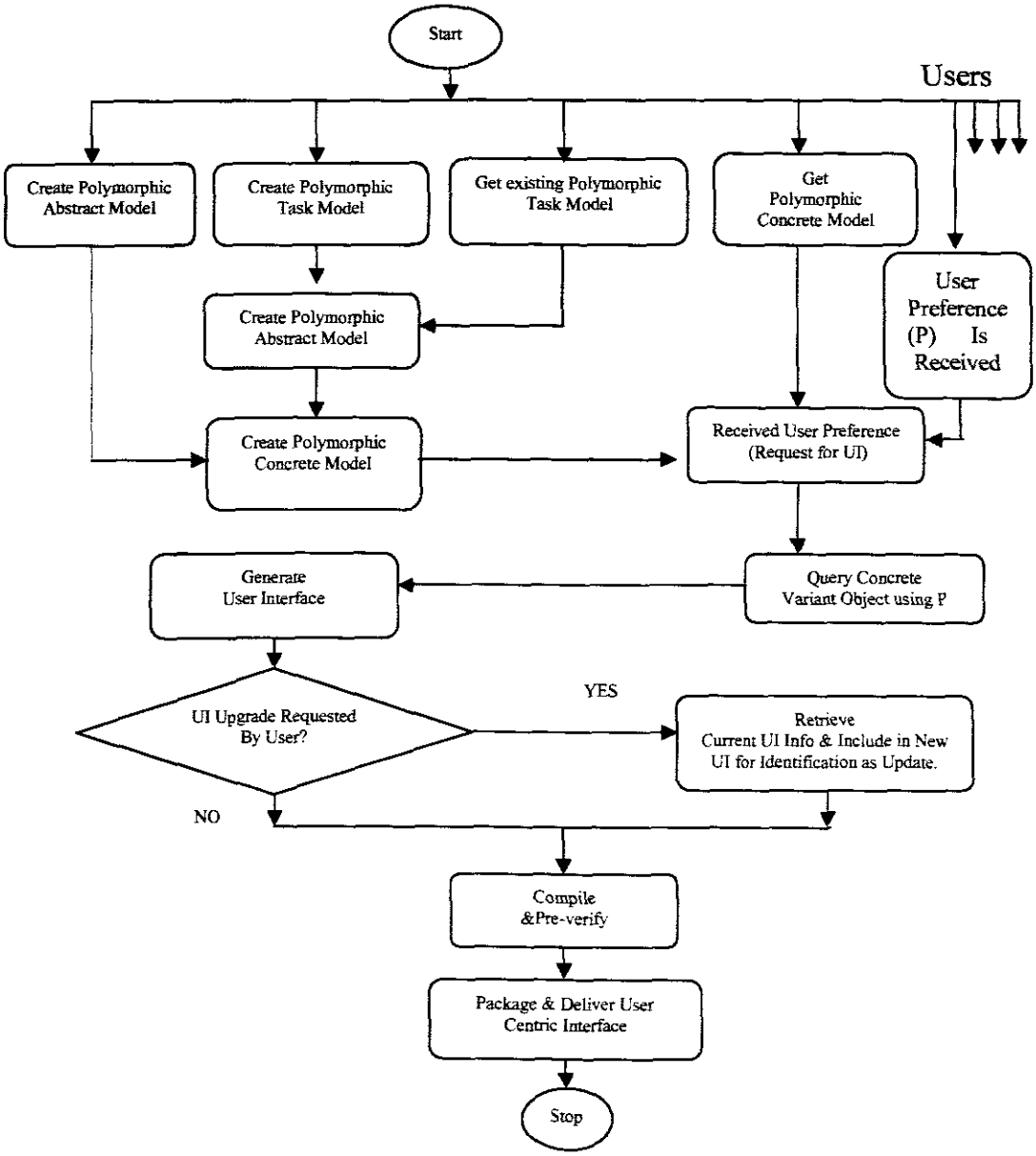
The runtime activities can also be considered as an extension to the load time activities. The activities during the runtime basically include modification and adaptation of an existing interface to changes in user preferences. In case where an adaptation of an interface is required by a user during runtime, user interface generation is based on a new set of user preferences. In addition, the Interface Adaptation Engines (IAE) retrieves from the Active User Repository (AUR), the description of the current interface on the requesting user device. The description includes interface identification information. The Packager includes the retrieved information with the generated interface during

packaging, prior its delivery to the user. During installation, the description information identifies the generated interface as an upgrade to an interface existing on a user device. The Application Management Software (AMS) of the J2ME enabled mobile devices identifies an upgrade-interface interface through the description information created by the packager. The AMS prompts the user for an upgrade to an existing user interface. Figure 4.3 presents a sequence diagram for the load and runtime activities of the proposed system. A description of the activities with the CoMADE toolkit is represented using an activity diagram (Figure 4.4).

4.3 The Implementation of CoMADE Toolkit

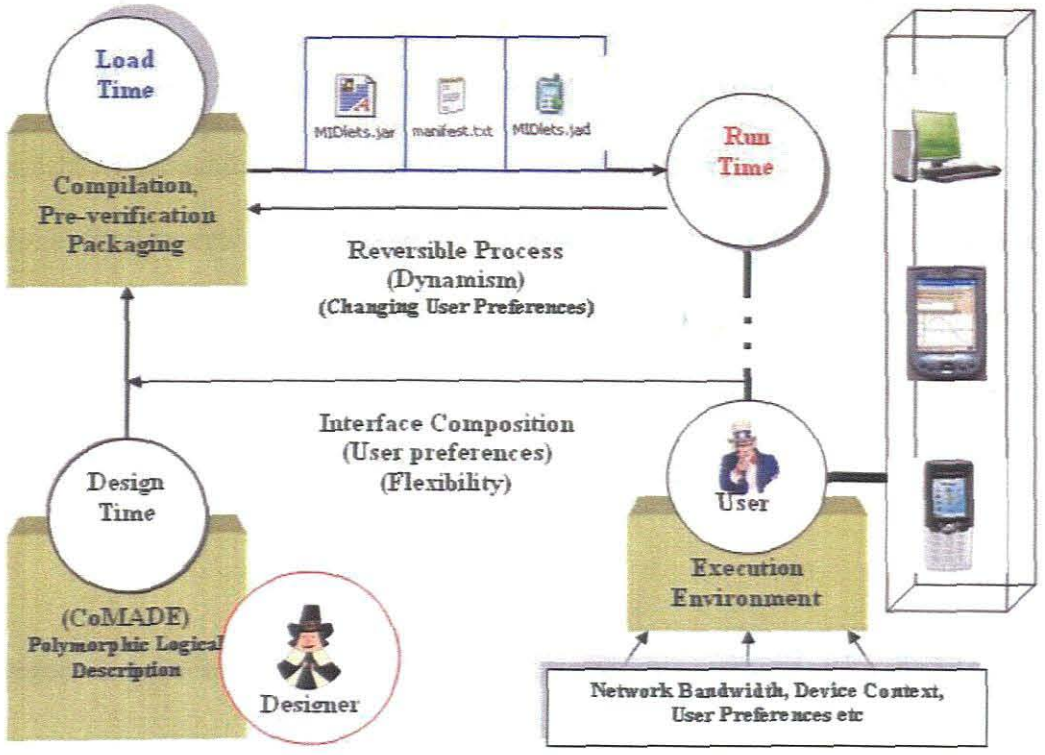
An authoring environment was implemented for the generation of multi-target user interface based on our iPLD framework. CoMADE was implemented in Java and it adopts the look and feel of most Integrated Development Environment (IDE). A mediator, example-preference submission section and emulator programs are integrated with CoMADE for design time testing. CoMADE is used by the designer for creation of polymorphic logical description during the design time. CoMADE generates interface (Java Classes (MIDlets)) for individual users. User interfaces are generated dynamically for each requesting user by the selection of CIOVariants. MIDlet classes are generated for every set of UPTS.

Figure 4. 4 Activity Diagram for the Proposed System



The MIDlets together with other non-midlet Java files are compiled and pre-verified prior the creation of manifest file, which contains a description of the MIDLet suite (Figure 4.5). The Mediator archives the generated files into a Java Archive File (JAR).

Figure 4. 5: Required Implementation Steps of CoMADE



Finally, an Application Descriptor (JAD) is created for every interface package. The JAD contains in addition to the content of the manifest file, the size of the MIDet suite and the URL of the jar file. A URL, which points to the location of the generated user interface presented to the user device. When the user clicks on the URL, the MIDlet suite (User Interface) is installed in accordance to the Over the Air (OTA) provisioning.

With these afore-outlined steps, each requesting user is provided with an individualized, user centered MIDlet suite, which is dynamically generated on demand.

Using CoMADE, designer can set some preferences, initiate the mediator and test the generated interfaces on some emulators integrated with the system, for example Sun

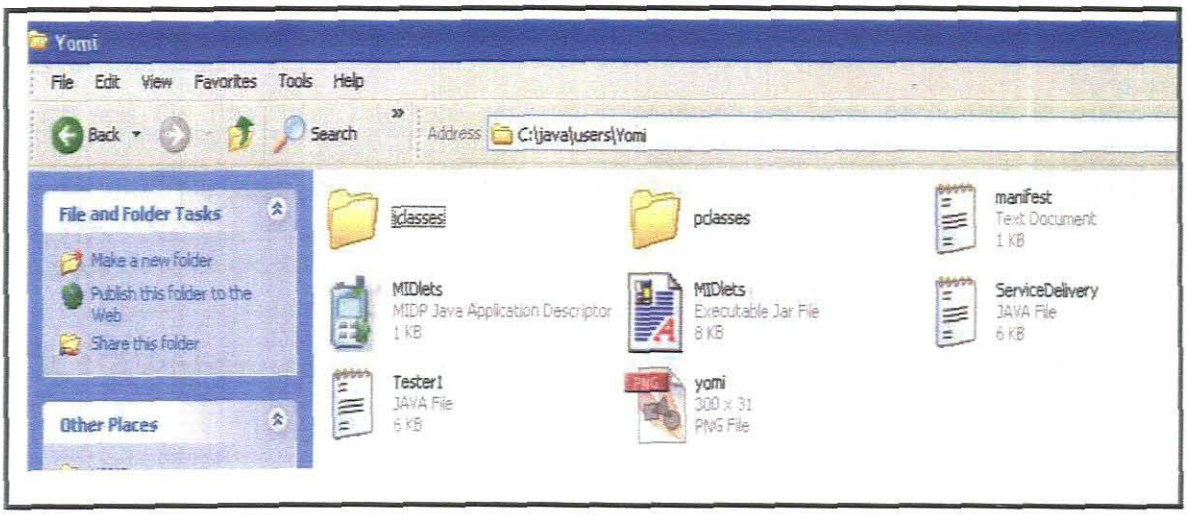
MicroSystem emulator programs were used for design time testing of sample applications and also, designers can extend the emulator programs already integrated with CoMADE.

Dynamic Compilation and Pre-verification is carried out by the Packager component of the mediator. The packager creates new directory for the users during the load time, the generated java classes for a requesting user are compiled and pre-verified prior generation of a URL for a requesting user. Figure 4.6 shows the compilation and pre-verification sample code while a sample directory of an interface resource files created dynamically for a requesting user is shown in Figure 4.7.

Figure 4. 6 Code for Dynamic Compilation and Pre-Verification by the Packager Component of the Mediator

```
String  filesForCompilation = "javac  -bootclasspath " + pathMidpClasses  +
                              "-d  " + jcfiles  +"  "+ userpath+ "\\*.java";

        Process process  = runtime.exec( filesForCompilation);

        process.waitFor();

        log.append("\n" +"..." +  "BootClassPath Compiling : Successful !!! " );


String filesToPreverify = "preverify -classpath " + pathMidpClasses +
                          " -d   " + pcfiles +"  "+jcfiles;

        process  = runtime.exec(filesToPreverify );

        process.waitFor();

        log.append(" n" +"..." + "Class Preverification Successful !!!"  );
```

Figure 4. 7: Interface Resource File



The snapshot source code in Figure 4.6 shows the dynamic compilation and pre-verification carried out by packager component of the mediator during an interface generation. Figure 4.7 shows an interface resource file directory created automatically in response to an interface request by a user. Sub-directories namely, the *jclasses* and *pclasses* are also created for each user. The jclasses folder stores the compiled java source files that are generated in response to a user interface request and pre-verified java classes are stored in the pclasses folder. As new requests for interfaces are received by the mediator, new directories are created for storing the generated interface files.

4.4 The CoMADE Environment

The various phases in the proposed PLD are represented differently on CoMADE. This Section therefore, presents the phases of PLD on CoMADE. In addition, the mediator

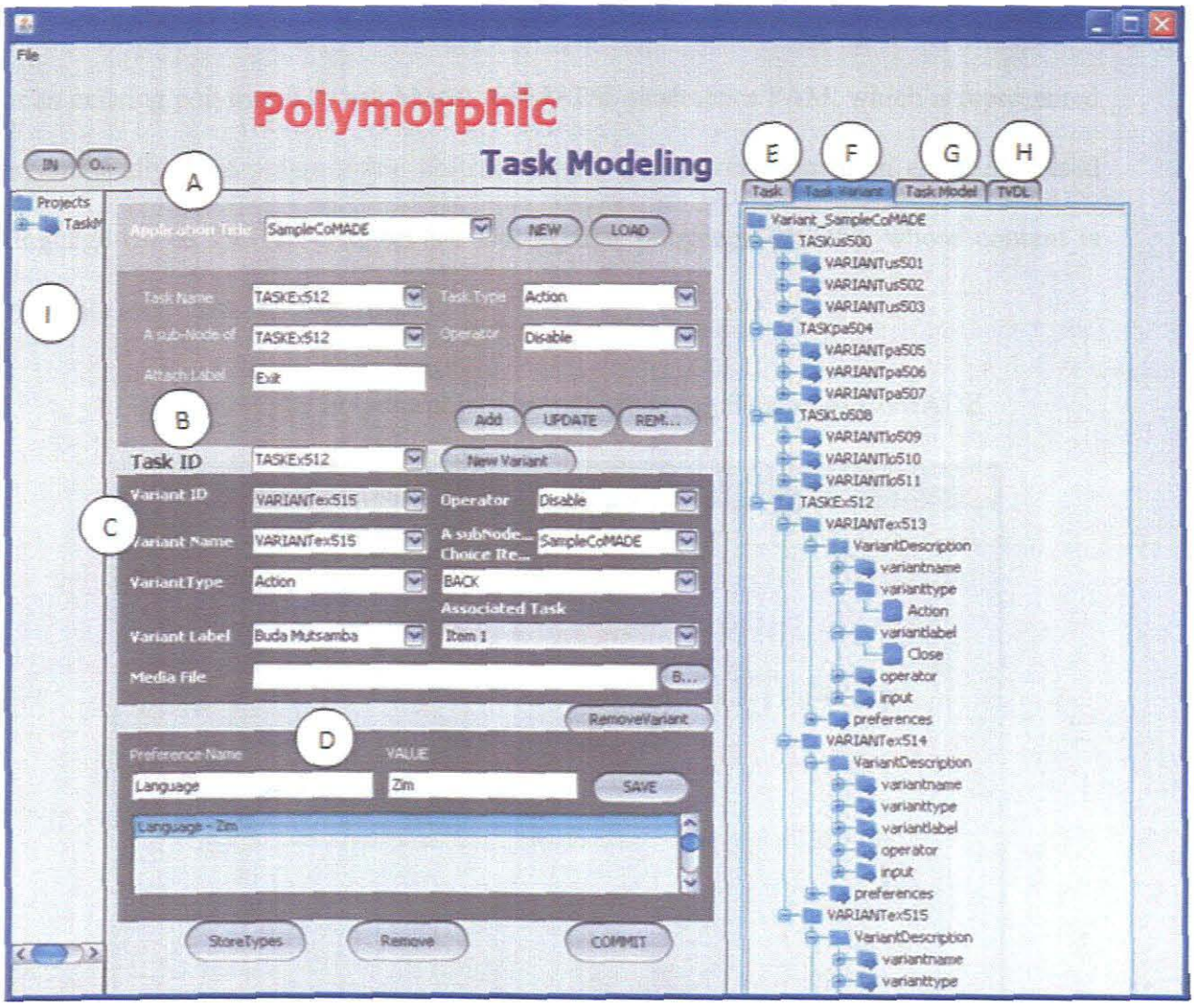
component for design time generation of an interface through sample preference specification is presented in this section. The polymorphic task modeling (Figure 4.8), the polymorphic abstract modeling (Figure 4.9), polymorphic concrete modeling (Figure 4.10) and the final user interface generation phases have been decoupled in order to permit multiple start points for designers. Designers can either create a new model or proceed from an existing one.

4.4.1 A Polymorphic Task Modeling Section

This phase is comprised of the task information specification, variant and preference part and a hierarchical structure of polymorphic task model.(Figure 4.8).

Figure 4.8 shows a Polymorphic Task Modeling section of the CoMADE toolkit and labels A to I represent some of its subdivisions. At the start of a new project, an interface designer provides the title of the project on **A**. The designer can proceed to specifying the tasks that users must perform on an interface in order to realize a goal. For example "Enter Your Username", "Enter Your Password", and clicking of "Login" can represent set of tasks that must be carried out by a user in order to have access to some services. Each task, for example, "enter your name" can be given alternative representations called variants. One or more variants can be associated with a task, which can be identified by a "Task ID". The properties of each variant are specified on C and D. Tasks and their associated properties are hierarchically specified in a tree structure on division E. Tasks with their associated variant description are represented hierarchically on division F.

Figure 4. 8: The Polymorphic Task Modeling Section of CoMADE.



An xml representation of the hierarchical task representation on division E is viewable on G. Also, the resulting polymorphic task model, realized from task and variant information specification in XML representation called Task variant Descriptor Language (TVDL) is viewable on H. Existing projects and models can be accessed on I.

4.4.2 A Polymorphic Abstract Modeling Section

The Polymorphic Abstract Modeling requires less of designer's input, upon specification of an existing polymorphic task Model, CoMADE generates a PAM, which is represented hierarchically. The polymorphic abstract model is also represented in an XML based language called the Presentation Set Description Language (PSDL), whose content is viewable on CoMADE (Figure 4.9).

Figure 4. 9: The Polymorphic Abstract Modeling Section of CoMADE

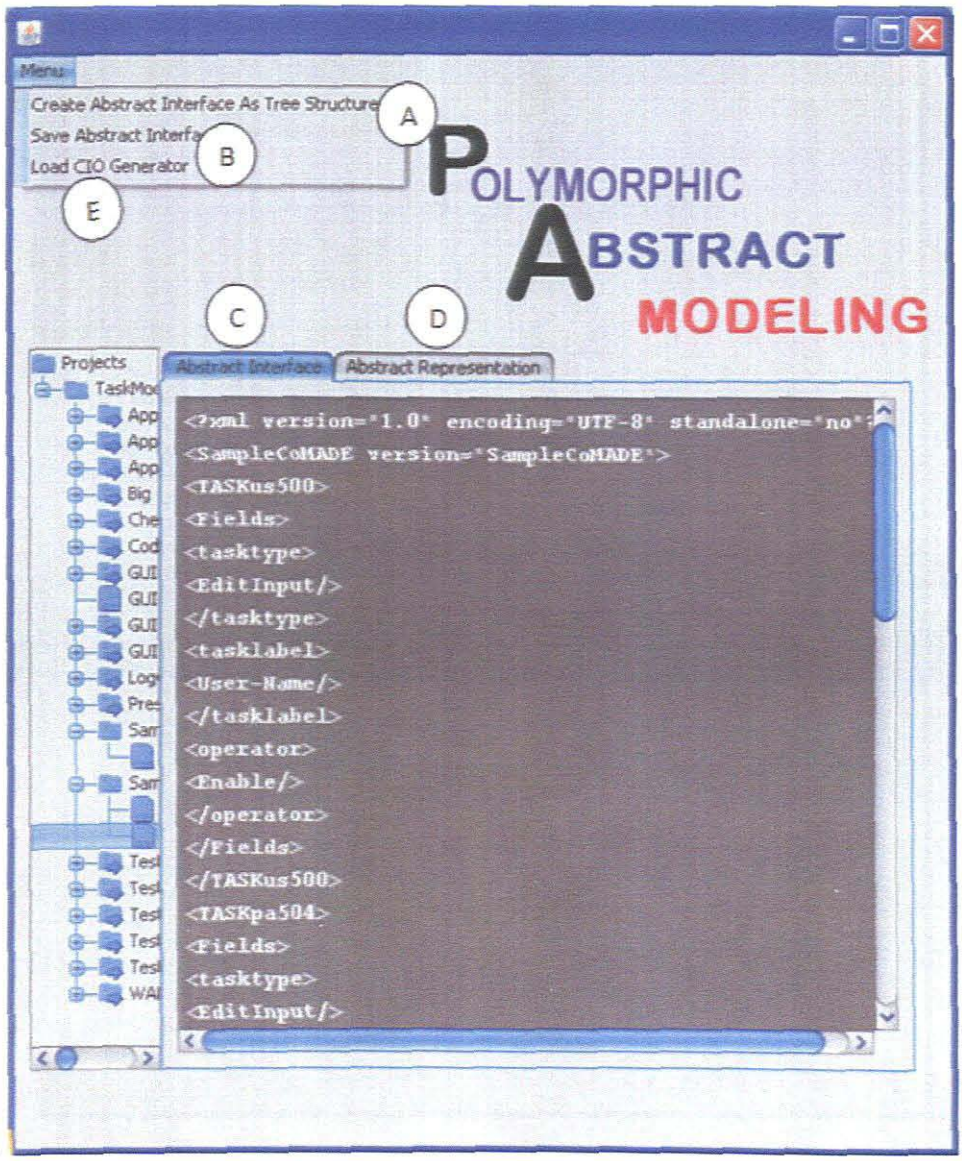
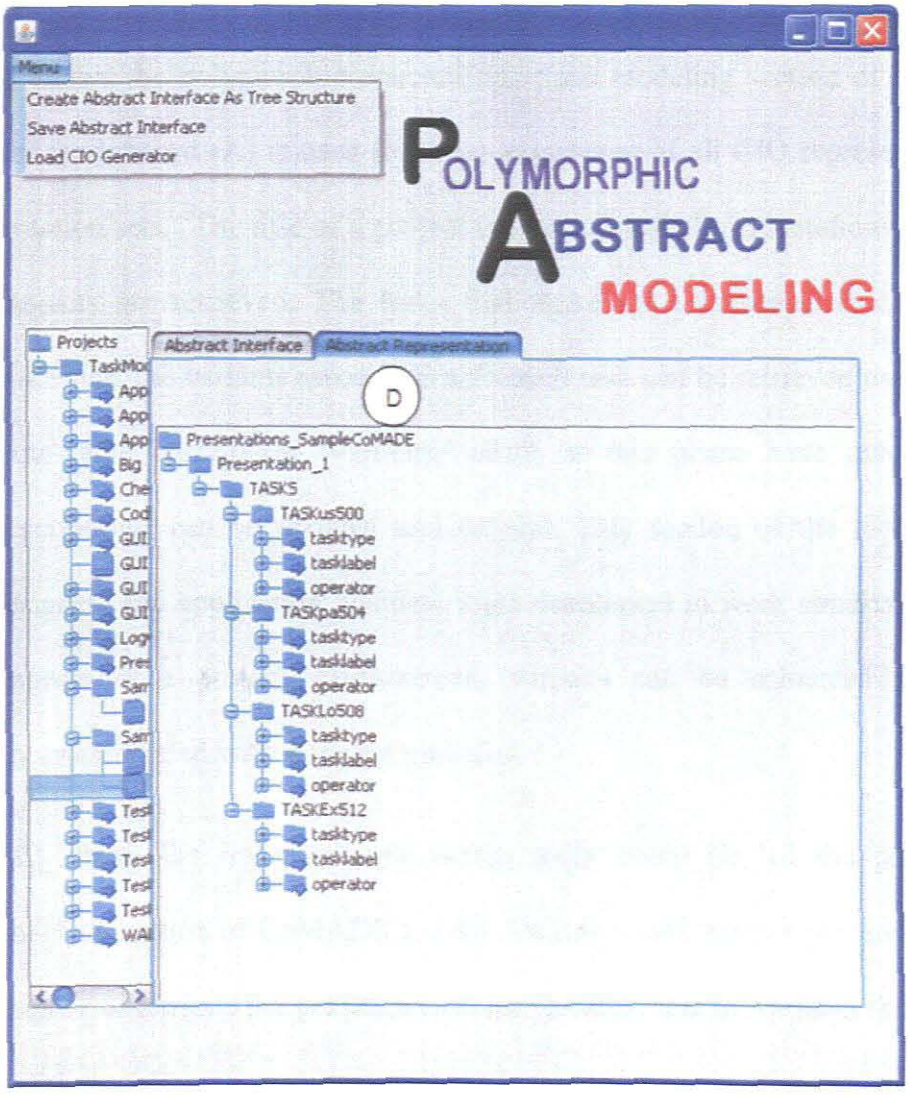


Figure 4.9 shows a Polymorphic Abstract Modeling section of the CoMADE toolkit and labels A to E represent some subdivisions of the interface. The next step after the creation of a polymorphic task model is to transform the model into a polymorphic abstract model (PAM) by the selection of the menu item labeled A. The resulting PAM is organized into different presentations and can be saved by clicking on B. The PAM is represented in an XML based language called the Presentation Set Description Language (PSDL) and also in a hierarchical structure as shown in C (Figure 4.9) and D (Figure 4.10) respectively.

Figure 4. 10 : Hierarchical Representation of Presentation Set Description Language



4.4.3 Polymorphic Concrete Modeling and Final User Interface Generation.

The Concrete model, which can also be referred to as concrete interface is generated at this stage. This phase allows for persistence of concrete representation of variant objects, modification of implementation methods (Java implementation methods) generated for each variant object and specification of example preference for generating final interfaces (Figure 4.11). The mediator can also be initiated in this section. The mediator generates and loads a sample interface on an emulator for testing and evaluation purposes.

Figure 4.11 shows the polymorphic concrete interface modeling section of CoMADE. The designer work board (A) is used for the management of all CIO representations of every presentation sets. The title of a project is selected and all presentations associated with the projects are retrieved. The tasks that make up a presentation can also be retrieved. Likewise, the variants associated with each task can be retrieved together with their attribute-value description. Variants, which at this phase have assumed their concrete descriptions, can be updated and deleted. This section of the toolkit allows different designers and application business logic developers to work simultaneously on different aspects of a project. Furthermore, variants can be enhanced with more functionality or associated with different modules.

Figure 4.12 shows the interface composition work board (B) of the polymorphic concrete modeling section of CoMADE toolkit. On this board, sample preference can be set by a designer to retrieve the preference-aware variants, that is, variants that meet the

specified sample preferences. Other components of an interface include the class template information, mundane methods or the application independent methods.

Figure 4. 11 :Polymorphic Concrete Modeling: The Designer Work Board (A)

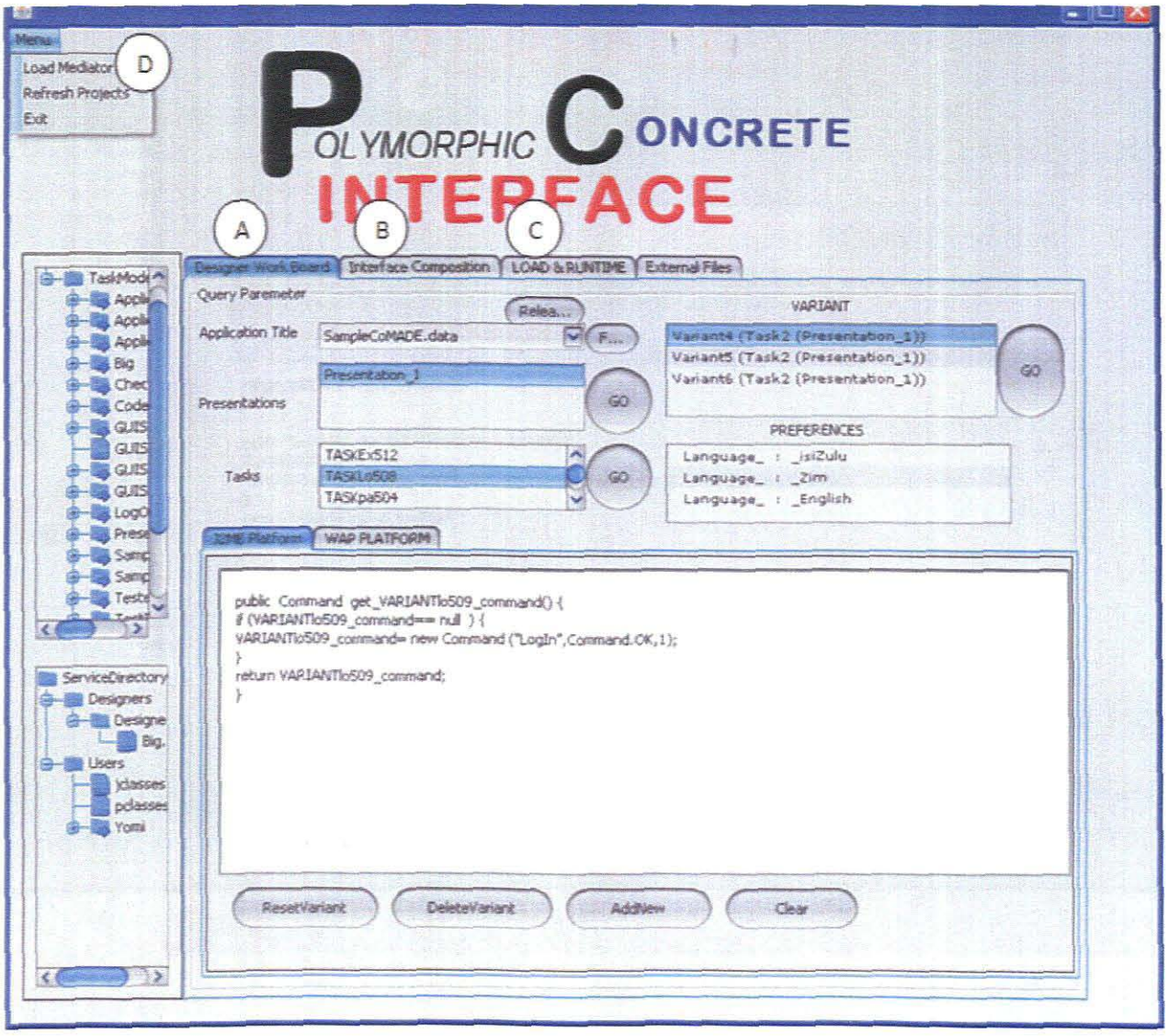
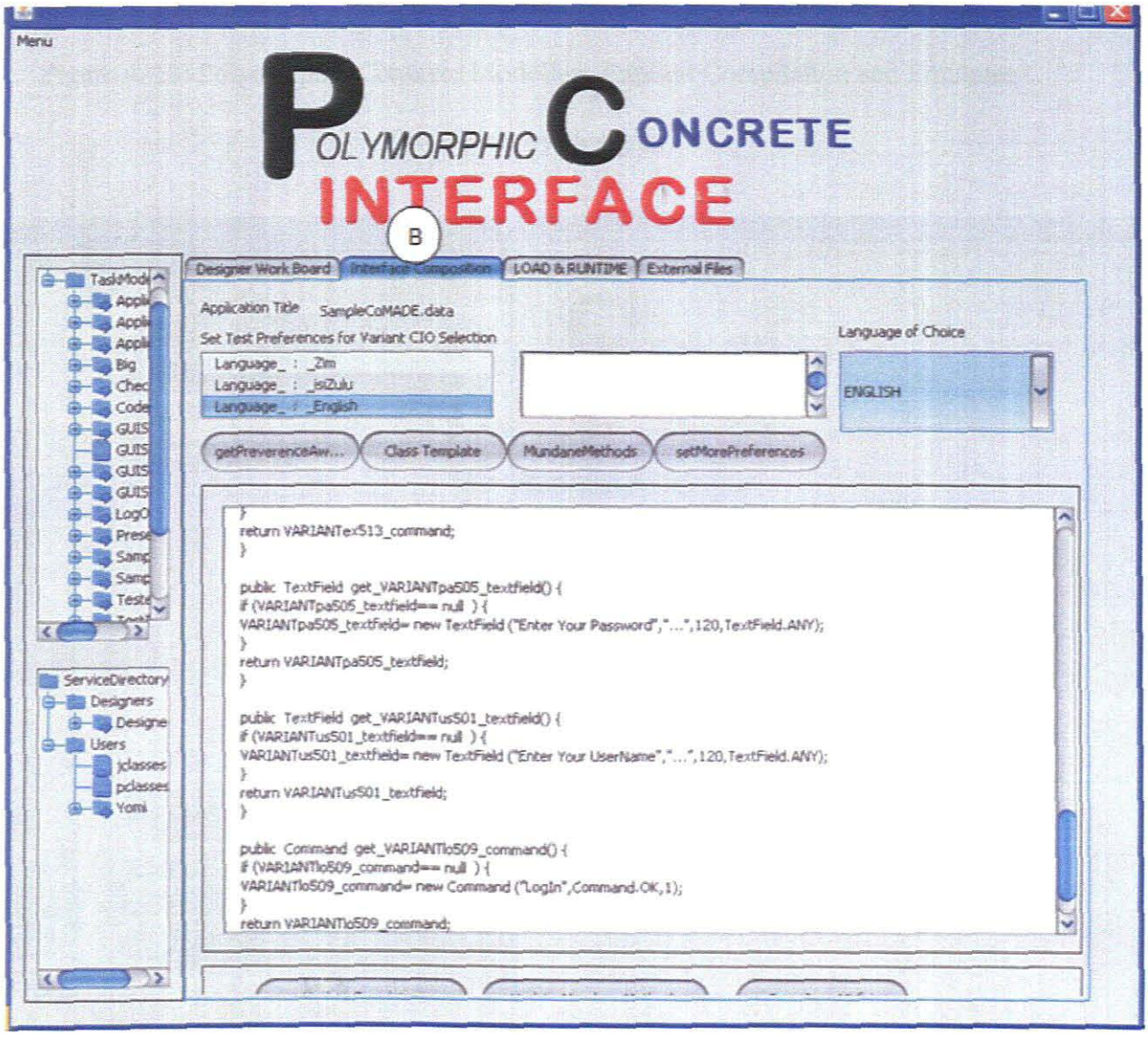


Figure 4. 12: Polymorphic Concrete Modeling: Interface Composition (B)



The building of an interface through selection of preference-aware variants and other components is called an interface composition. Interfaces can be composed and loaded onto an emulator program either by explicit interface composition and compilation by a

designer (Figure 4.13) or by the use of the mediator agent, which automates the interface composition process (Figure 4.14).

Figure 4. 13: Polymorphic Concrete Modeling: Explicit Compilation and Emulator Invocation by Designer

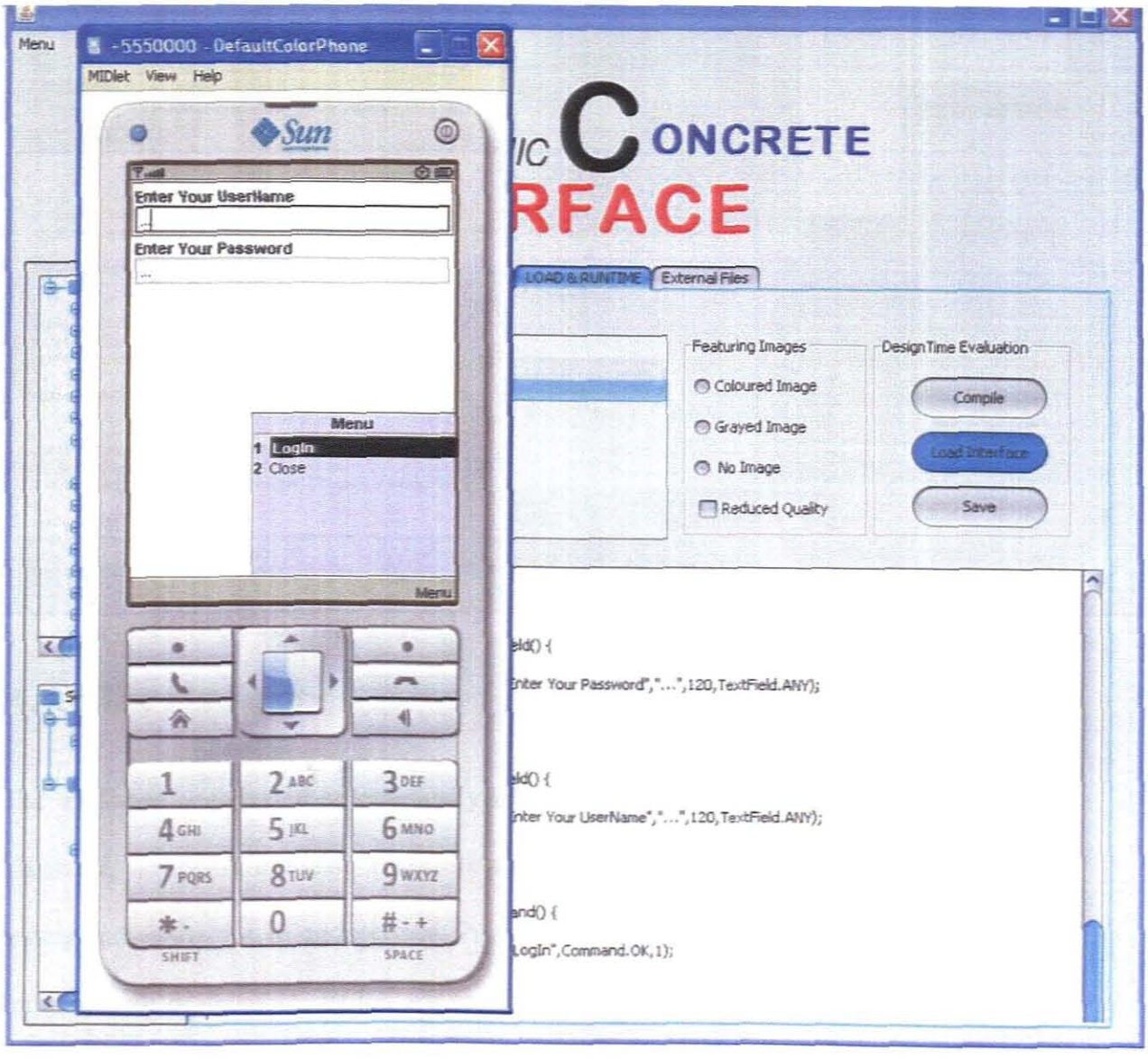
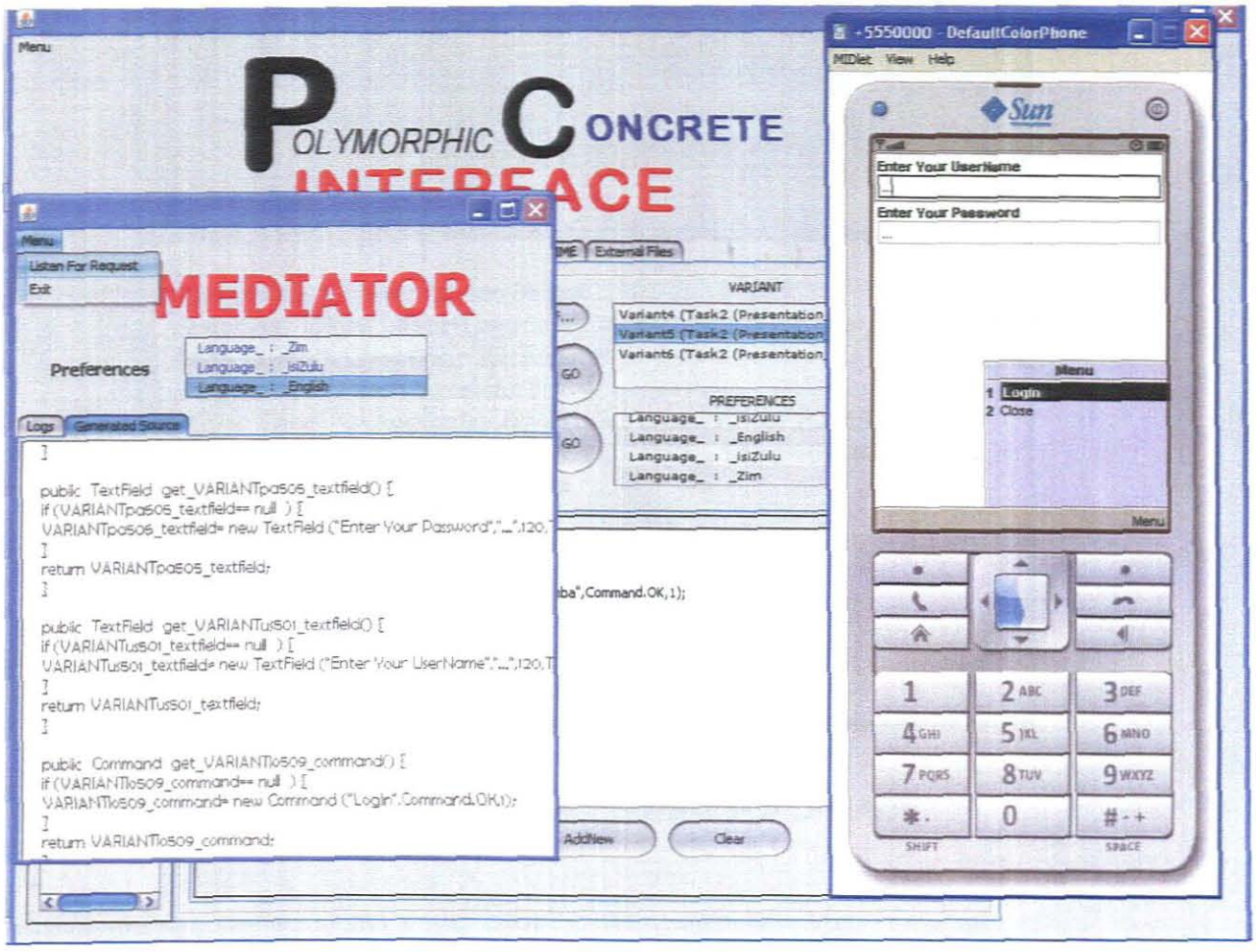


Figure 4. 14 The Polymorphic Concrete Modeling: Automatic Interface Generation by Mediator



4.4.4 The Mediator

The mediator is activated automatically when a request is received from the end-user. Likewise, designers can also manually activate the mediating agent for design time testing and evaluation of an interface. Prior the design time testing, designer can also set some sample preferences, which are used in the generation of a final interface. Figure 4.15 is a graphical interface of the mediator showing a log data of a recently generated interface.

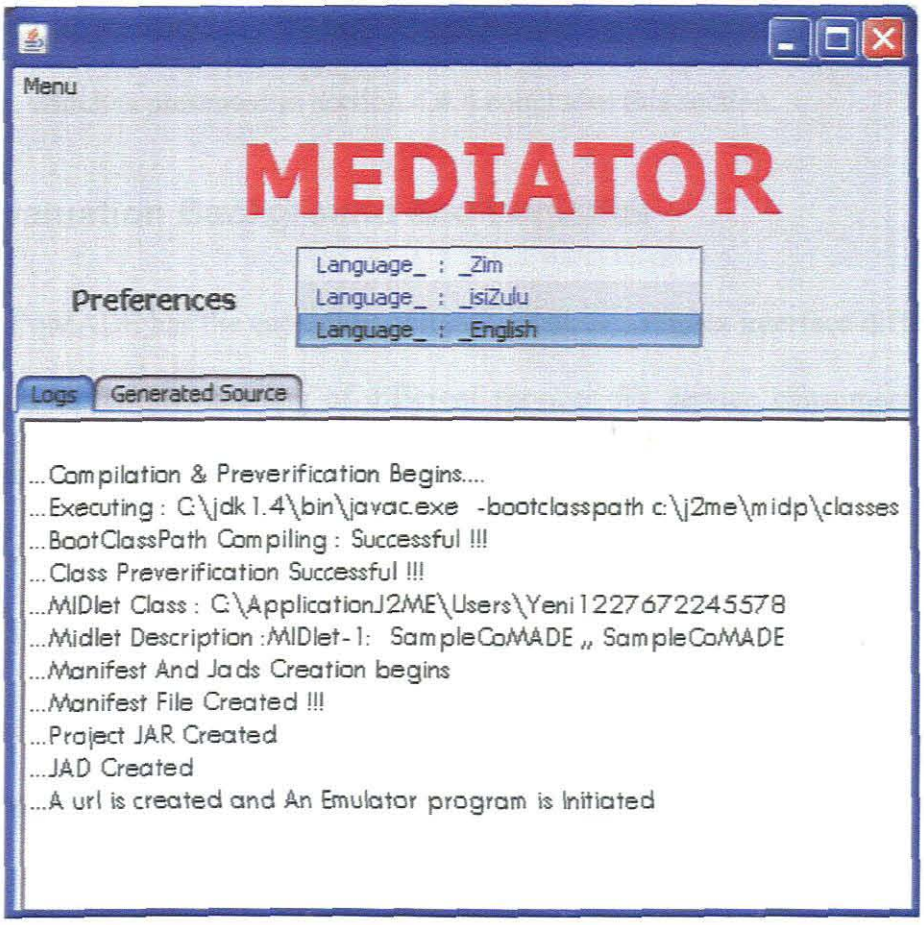


Figure 4.15. the Mediator Integrated Into CoMADE

4.5 The Evaluation of the Proposed System

This section presents the evaluation of the prototyped toolkit and the usability of generated interfaces. CoMADE is based on iPLD framework. The evaluation was carried out according to procedures for usability testing proposed by (Nielsen, 2000). Firstly, the background and logistics to the evaluation is presented in section 4.4.1. Secondly, section 4.4.2 gives the evaluation procedures followed in this study. Section 4.4.3 presents the

user interfaces generated by CoMADE toolkit during evaluation. The result of user evaluation, which is presented in section 4.4.4 concludes this section.

4.5.1 Evaluation Background and Logistics

Although CoMADE has the capability of handling more complex interface differentiation levels, such as the provisioning of different services for service consumers based on service request and preference information, we have chosen a simple but adequate example for the purpose of clarity, evaluation and demonstration of the practicability of the toolkit. The proposed system was evaluated based on the following:

I. The effectiveness of CoMADE in generating a Polymorphic logical description method is evaluated based on the following abilities:

   a. Design of polymorphic task model

   b. Automatic Generation of Polymorphic Abstract Model

   c. Automatic Creation of Polymorphic Concrete Model.

   d. Generation of Final User Interfaces.

II. Generation of User Interfaces in accordance to specified preferences.

   a. Ability to generate an interface to meet a set of specified user preferences, which include choice of media content, choice of language and choice of service-usage reporting style.

   b. Ability to adapt an initially generated interface to a new set of user preferences.

III. The required time for generating an interface with CoMADE.

IV. The ease of use of the CoMADE Environment.

The evaluation was carried out in a computer laboratory with 14 inches screen sized computer systems. Three categories of testers were considered for the evaluation, namely the expert (E), the intermediate (I) and Novice (N) user interface designers. We considered thirty (30) testers in all, with ten testers in each group, although Five (5) were proposed as adequate for usability (Nielsen, 2000). The testers acted as both interface designers and service users (see Appendix for questionnaire). The first questionnaire was used after the evaluation for classifying testers into their respective classes based on their responses. A second questionnaire was administered to testers after having a practical design section with CoMADE. The questionnaire was for capturing testers' opinions about the claims of CoMADE and the usability of the generated interfaces.

4.5.2 Evaluation Procedure

The testers were given a brief introduction about the procedure for the evaluation. We introduced the testers to CoMADE environment and its use through a video-presentation. The testers were each given an application interface, titled "ServiceUsageReport" to design with the CoMADE toolkit. The application allows service providers to query the usage of their services deployed on a grid infrastructure. Service providers can generate service report in any chosen format, histogram, barChart, tables are some of the options. During polymorphic task modeling, testers associated different attribute-values with task variants. In addition to the consideration of alternative styles of report generation, different forms of media content adaptation were considered and different languages

were specified as task variants to address the diverse languages in KwaZulu-Natal province of South Africa.

4.5.3 The CoMADE Generated User Interfaces

Due to the attribute-value descriptions given to different variants, different interface presentations were achieved. Testers specified the language of choice and the choice of report presentations. In addition, the testers specified image adaptation choices, which included colored or grayed images. These specifications are attribute values associated with task variants during the polymorphic task modeling phase.

During evaluation, different example-preferences were supplied by testers. CoMADE generates some interface presentations in English Language with colored images and tables for service usage report presentation (Figure 4.16). Some user interfaces composed of grayed images and tables were used for report generation (Figure 4.17).

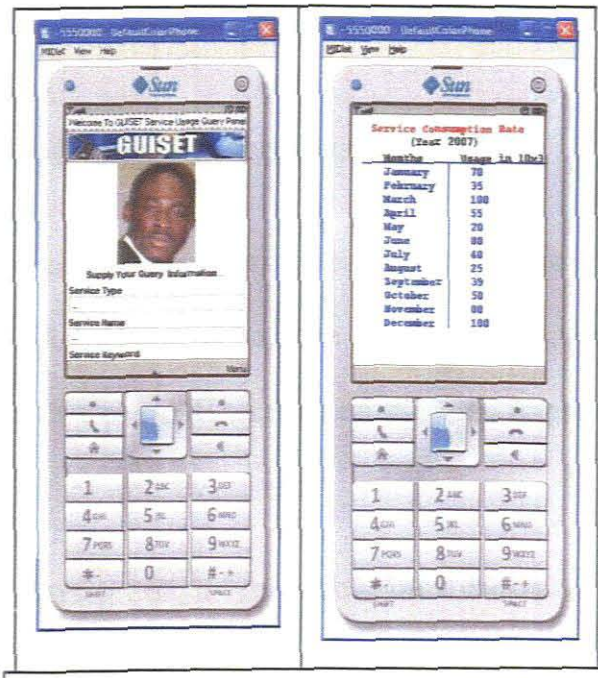


Figure 4. 17: User's Choice - Lang: English, Image: Coloured , Report: Tables

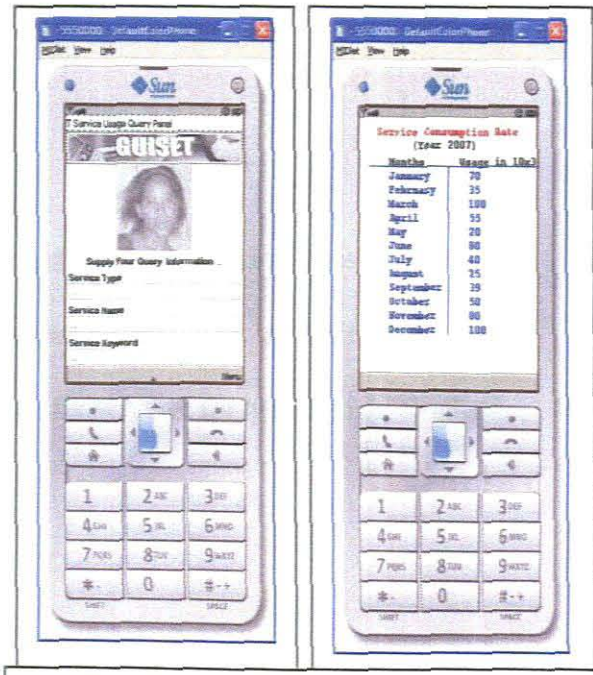


Figure 4.16: User's Choice - Lang: English, Image: grayed, Report: tables

User interfaces were also generated in isiZulu language. Some user interface presentations were achieved without an image as specified in the example-preferences (Figure 4.18). All generated user interfaces were according to the specified preferences.

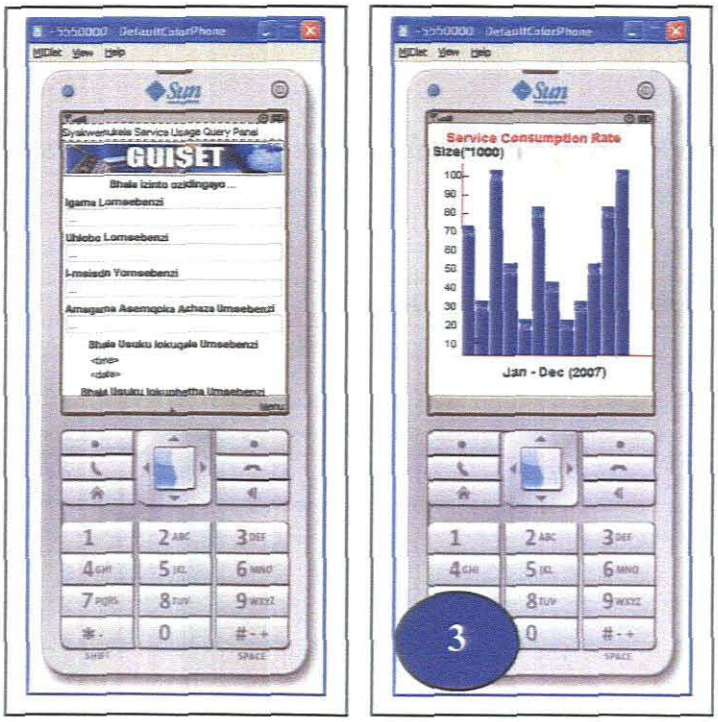


Figure 4. 18: User's Choice - Lang: isiZulu, Image : Non, Report: Chart

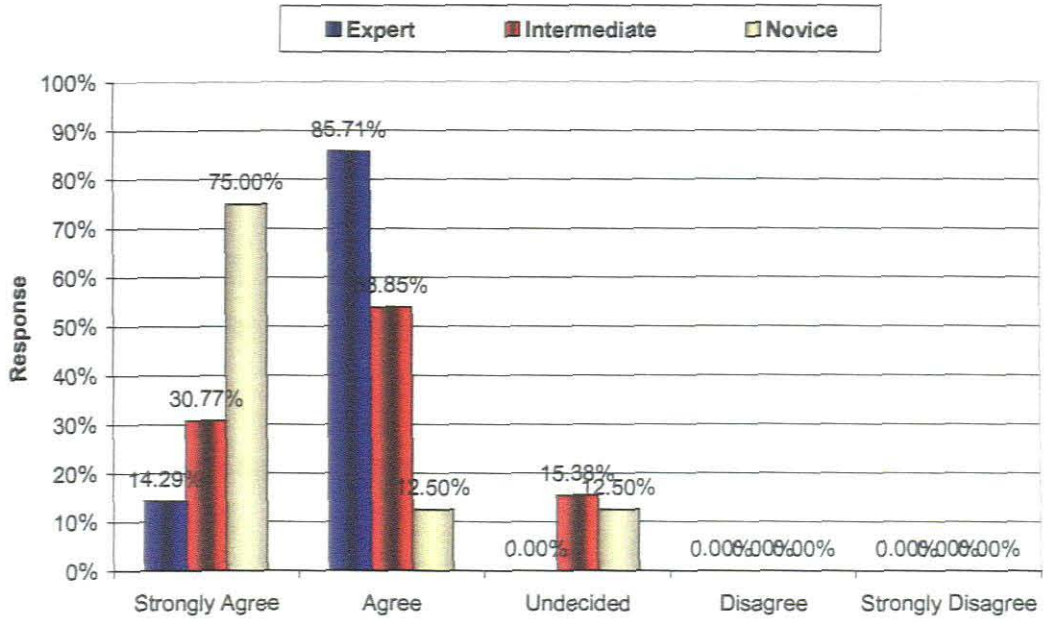
4.5.4 Evaluation Results

This section presents the analysis of the data collected using the second questionnaire. As highlighted above, both the design time modeling, runtime activities including the ease of use of CoMADE toolkit and the time required by CoMADE for generating final user interface were evaluated.

A. Design of polymorphic task models

Figure 4.19 shows that 14.29%, 30.77%, 75% of expert, intermediate and novice strongly agreed that CoMADE supports the design of polymorphic task model. Similarly, 85.71%, 53.85%, 12.50% of expert, intermediate and Novice agreed that CoMADE supports the design of polymorphic task and 15.38%, 12.50% of intermediate and novice could not decide.

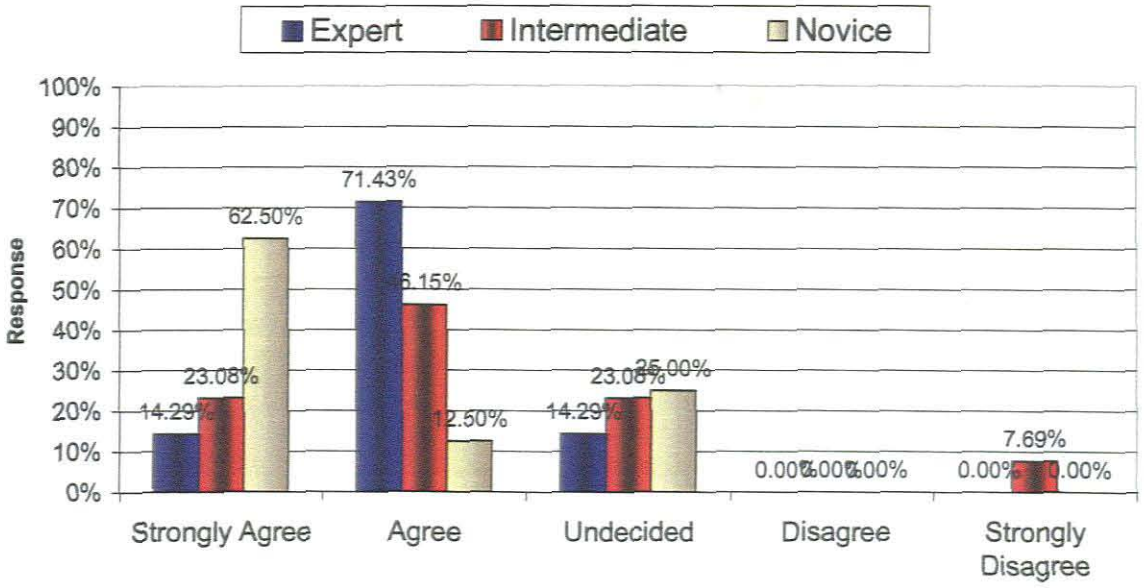
Figure 4. 19 Evaluation of CoMADE's ability for the design of Polymorphic task model



B. Automatic Generation of Polymorphic Abstract Model

Figure 4.20 shows 14.29%, 23.08% and 62.50% of the expert, intermediate and novice class strongly agreed that CoMADE can generate polymorphic abstract model, whereby each task variant is represented in terms of an abstract interaction object. Similarly, 71.43%, 46.15%, 12.50% of expert, intermediate and novice agreed that CoMADE can generate a polymorphic abstract model and 7.69 of the Intermediate Strongly disagree.

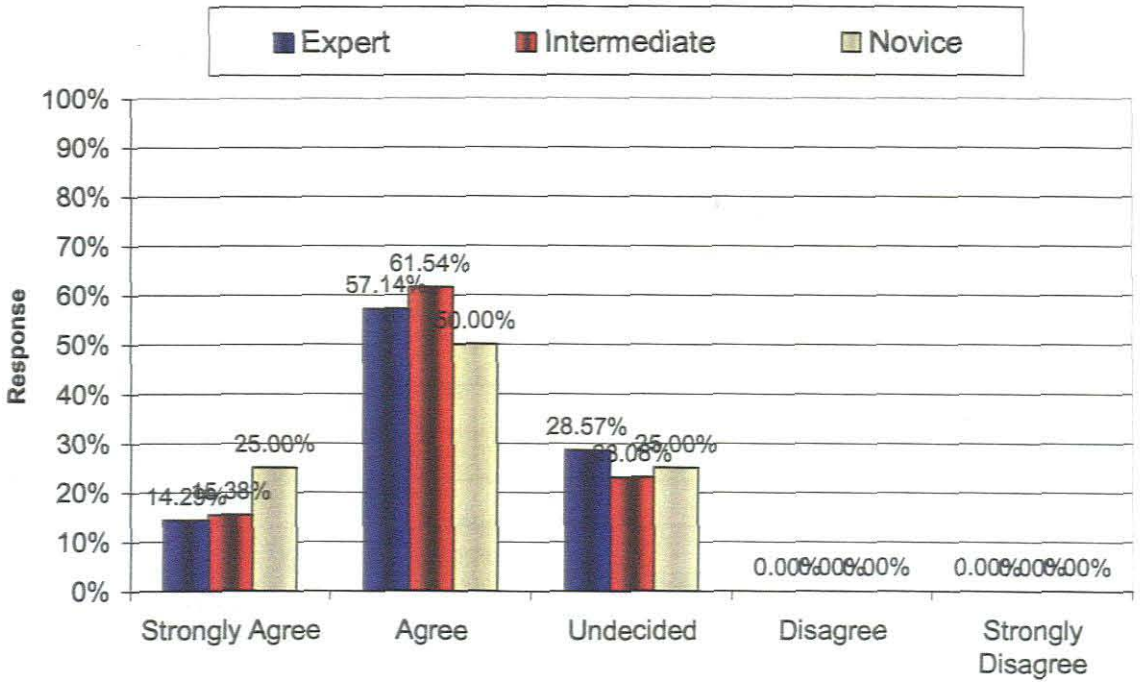
Figure 4. 20: Evaluation of CoMADE's ability in automatic generation of Polymorphic abstract model



C. Automatic Generation of Polymorphic Concrete Model

The ability of CoMADE in the generation of polymorphic Concrete Model was also evaluated and its ability to automatically persist the generated the Concrete Interaction Objects into an object database. Figure 4.21 shows that 14.29%, 15.38%, 25.00% of the expert strongly agreed that CoMADE supports dynamic creation and persistence of CIO variant implementation details. 57.14%, 61.54%, 50.00% of expert, intermediate and novice agreed that CoMADE supports dynamic creation and persistence of CIO variant implementation details. 28.57, 23.08%, 25.00% of expert, intermediate and novice could not make a decision.

Figure 4. 21: Evaluation for CoMADE's Ability for Automatic Generation of Polymorphic Concrete Model



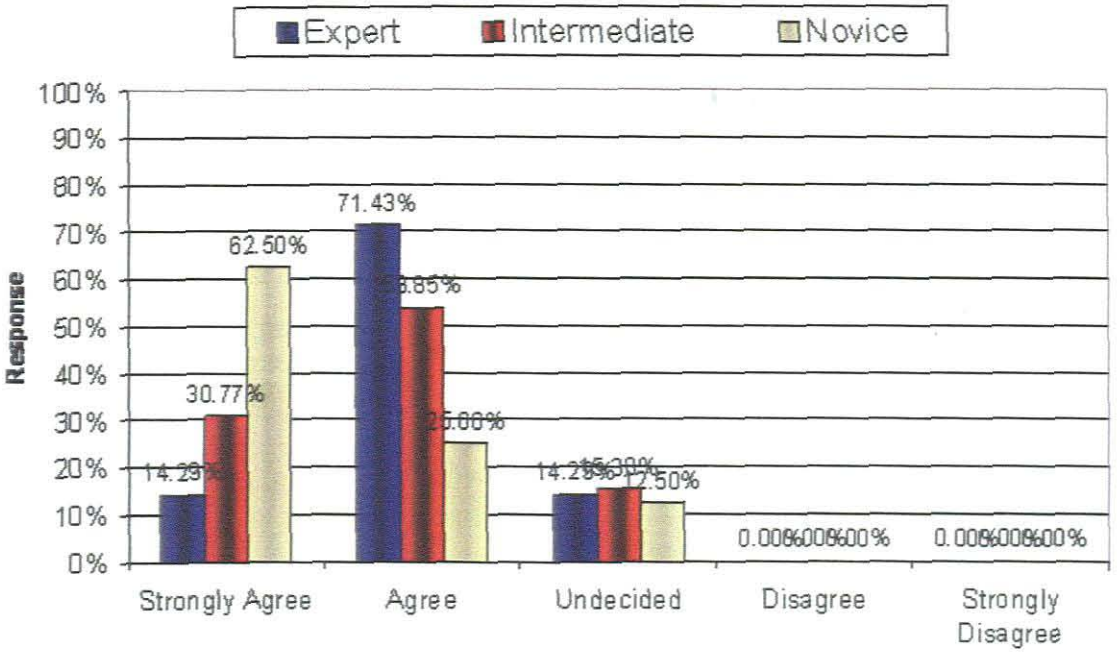
D. Evaluation of CoMADE's ability in the Variant Selection and Automatic generation of Final User Interface.

The ability of CoMADE to generate the final user interface was evaluated. The final user interface generation was carried out without a specification of example preferences.

Figure 4.22 shows that 14.29%, 30.77%, 62.50% of the expert, intermediate and novice strongly agreed that CoMADE is effective in the generation of the final user interface. Similarly, 71.43%, 53.85% and 25.00% of expert, intermediate and Novice agreed that

CoMADE is effective in the generation of the final user interface. 14.29%, 15.38% and 12.50% of expert, intermediate and novice could not make a decision.

Figure 4. 22: CoMADE's ability for automatic generation of Final User Interface

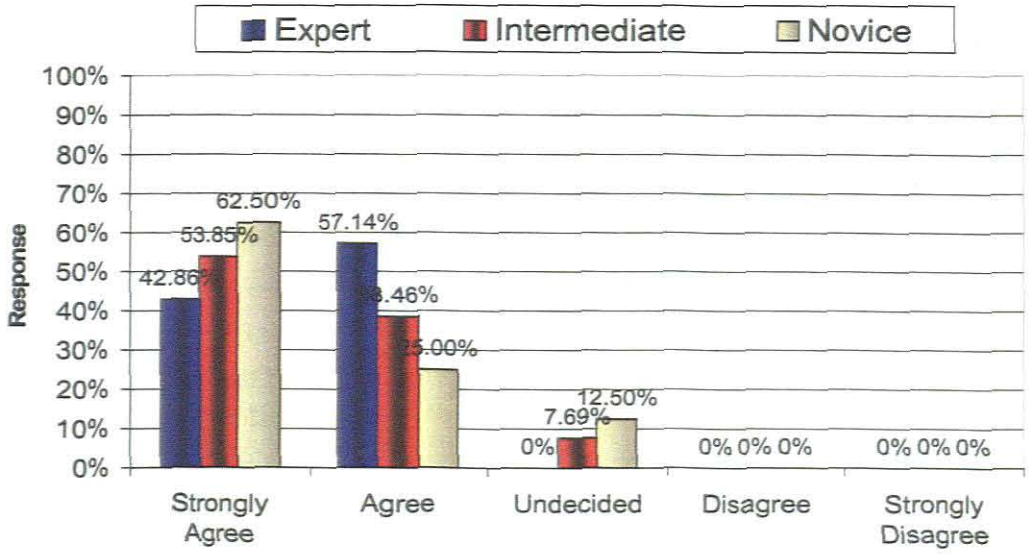


E. Automatic Generation of Interface Based On Specified Preferences.

The ability of CoMADE in the generation of user interface based on specified preferences was evaluated. This measures the ability of CoMADE to carry out interface composition. The interface composition involves the selection of Concrete Interaction Objects based on an example preference, the selection of application independent methods, class templates and the compilation, pre-verification and packaging of the interface. Figure 4.23 shows that 42.86%, 53.85%, and 62.50% of expert, intermediate and novice strongly agreed that CoMADE generates user interfaces according to the

specified user preferences. Similarly, 57.14%, 38.46% and 25% of expert, intermediate and novice agreed that CoMADE generates user interfaces according to the specified user preferences, but 7.69% and 12.50% of intermediate and novice could not decide.

Figure 4. 23: Evaluation of CoMADE in the generation of an interface based on specified preference

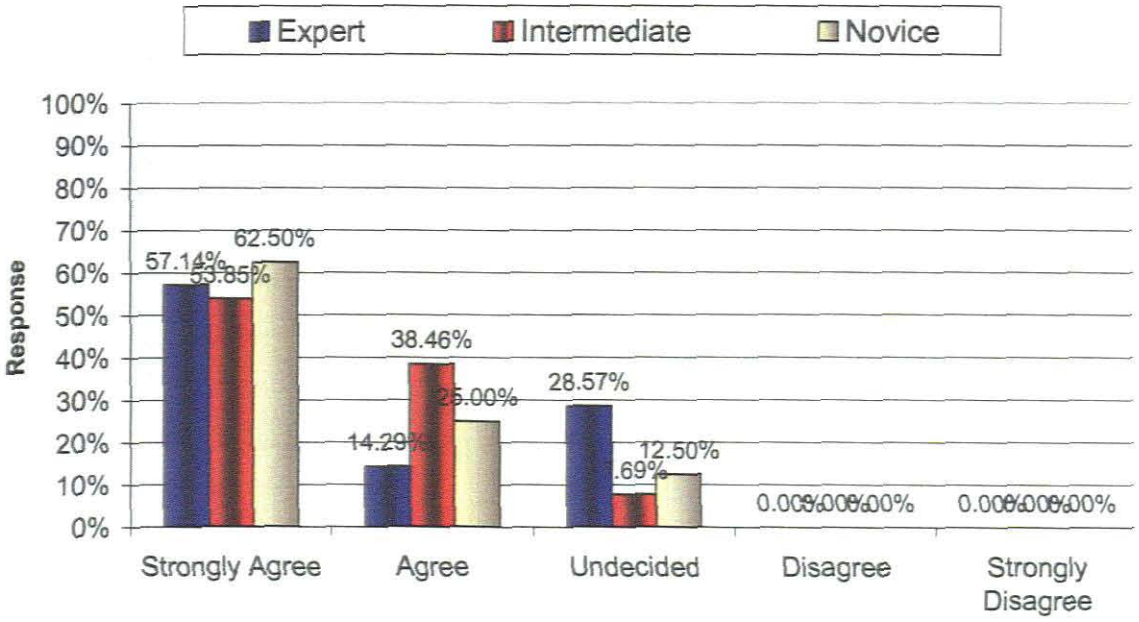


F. Adapting an existing or originally generated interface to a new set of Preferences.

This measures ability of CoMADE in adapting an existing interface to a new set of preferences. The preferences can either be an entirely new or a hybrid of a partly old or new set of preferences. In Figure 4.24, 57.14%, 53.85%, 62.50% strongly agreed that CoMADE has the ability to adapt an interface in response to an updated set of references.

14.29%, 38.46%, 25% of expert, intermediate and novice agreed that CoMADE can adapt an interface to an updated set of preferences and 28.57%, 7.69%, 12.50% of expert, intermediate and novice could not decide.

Figure 4. 24: Evaluation of CoMADE effectiveness in adapting an already generated interface to changes in preferences

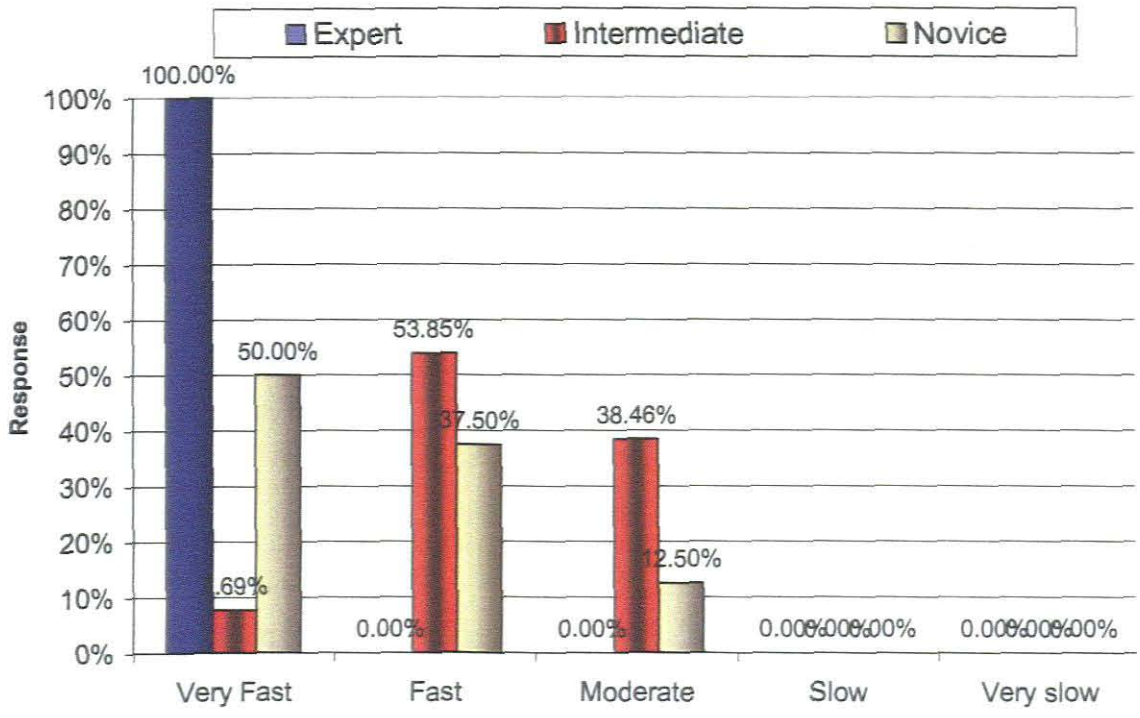


G. Response of Testers on how fast CoMADE generates a multi-target user interface.

Figure 4.25 shows that 100%, 7.69%, 50% of expert, intermediate and novice claimed that CoMADE is very fast in generating a multi-target user interface. Likewise, 53.85% and 37.50% of intermediate and novice claimed that CoMADE is fast in generating

multi-target user interface. Finally, 38.46% and 12.50% of intermediate and novice claimed that CoMADE is moderate in generating multi-target user interface.

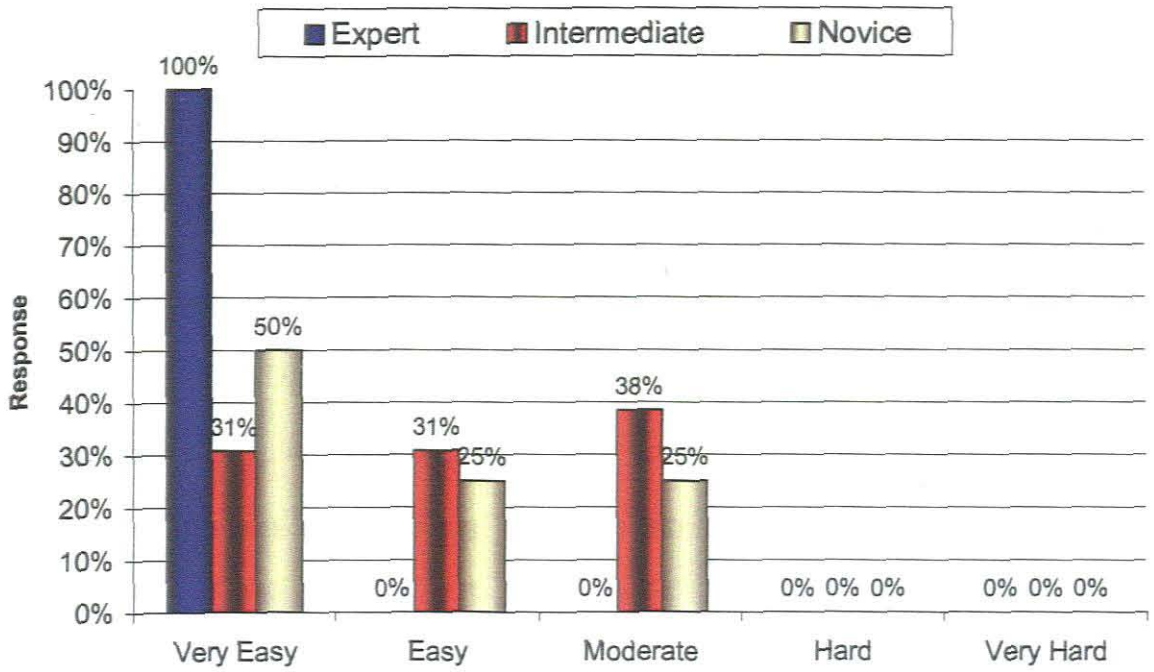
Figure 4. 25: Response to the speed of CoMADE in multi-target user interface design



H. Result of the Evaluation of the ease of use of CoMADE

The authoring environment was evaluated based on the ease of use, in order words; CoMADE was evaluated based on its user friendliness. Figure 4.26 shows that 100%, 31% and 50% of expert, intermediate and novice claimed that CoMADE is very easy to use. Similarly, 31% and 25% of the intermediate and novice claimed that CoMADE is easy to use, but 38% and 25% of intermediate and novice could not decide.

Figure 4. 26: Response to ease of use



4.6 Discussion of the Result

The result of the evaluation shows the ability of CoMADE in automatic generation of Polymorphic task model, Polymorphic Abstract Model and Polymorphic Concrete model, which are preference-aware models of PLD proposed for realizing a final multi-target user interface. With the CoMADE toolkit, an interface can be generated automatically according to a set of specified user preferences. CoMADE achieves automatic generation of an interface for individual user. CoMADE, gives priority to user needs and requirement during an interface generation. In addition, CoMADE can adapt an initially

generated interface to changes in preferences. Moreover, the CoMADE toolkit is easy to use.

4.7 Summary

In summary, this chapter describes the prototype design, implementation and evaluation of the proposed toolkit. CoMADE, which is based on the iPLD framework achieves direct user participation in multi-target user interface generation. The result of the evaluation of CoMADE shows its unique capabilities. Most importantly, its ability to initially generate user interface according to user pre-recorded preference information and also adapt user interfaces to changing user preferences.

The implementation and the evaluation of CoMADE have been presented in this chapter. The objectives of this study, which include the creation of polymorphic logical description, development of modeling algorithms, development of iPLD framework and a prototyping of a Custom-MADE toolkit based on the framework have been achieved.

generated interface to changes in preferences. Moreover, the CoMADE toolkit is easy to use.

4.7 Summary

In summary, this chapter describes the prototype design, implementation and evaluation of the proposed toolkit. CoMADE, which is based on the iPLD framework achieves direct user participation in multi-target user interface generation. The result of the evaluation of CoMADE shows its unique capabilities. Most importantly, its ability to initially generate user interface according to user pre-recorded preference information and also adapt user interfaces to changing user preferences.

The implementation and the evaluation of CoMADE have been presented in this chapter. The objectives of this study, which include the creation of polymorphic logical description, development of modeling algorithms, development of iPLD framework and a prototyping of a Custom-MADE toolkit based on the framework have been achieved.

CHAPTER FIVE

CONCLUSION AND FUTURE WORK

5.0 Introduction

The goal of this research was to develop interface modeling algorithms and toolkit for the design of multi-target interfaces that allows direct user participation in the composition and adaptation of interfaces. In a bid to achieve the stated goal, we proposed a polymorphic logical description model, developed interface modelling algorithms and framework for integrating our model with a mobile computing environment. We also prototyped a toolkit based on our proposed framework. The PLD model creates a preparatory ground for diverse user needs and preferences. It provides alternative representation for tasks on a user interface at the design time of an application. The interface modeling algorithms aid the creation of the PLD model. Moreover, our proposed framework achieves integration of the PLD with a mobile computing environment. The framework provides the infrastructure to integrate the PLD model of the design time with the load time and runtime activities. The infrastructures also aid direct user participation during interface generation. Section 5.2 of this chapter presents a summary of our contribution and Section 5.3 presents the conclusion of this study. Section 5.4 gives some directions for future work.

5.1 Summary of Contribution

Some of the major contributions of this study have been submitted and accepted for publication. The proposed model provides a multi-target user interfaces for providers and consumers of a pervasive grid environment (Ipadeola, et al., 2008**a**). In addition, our proposed model and its associated modeling algorithms achieve automatic generation of user centric interfaces (Ipadeola, et al., 2008**b**). Our CoMADE toolkit also achieves dynamism, flexibility, ease of application extensibility, ease of use and supports design time evaluation of generated interfaces. It overcomes the challenges associated with separate design concept for addressing the diversity in applications and services and achieves direct user participation during user interface generation (Ipadeola, et al.,2008**c**).

Finally, this study proposed and defined Interface Composition (IC) as a mechanism for dynamic generation of user interfaces. Specifically, we reported IC as an efficient mechanism for automatic generation of interfaces for J2ME Enabled devices (Ipadeola, et al., 2008**d**).

Hitherto, the goal of this study, which was to develop user interface modeling algorithms and tools for the design of multi-target interfaces that achieves direct user participation in the composition and adaptation (generation) of interfaces, has been achieved.

A. Benefits of our framework

The proposed framework offers significant benefits to end-users and interface designers alike. The proposed method, unlike the existing methods presented in (Mori, et al.,

2004a ; Bisignano et al., 2006) caters for the diversity in user needs and preferences during the design time and also have capability to automatically support new set of user preferences. The model offers a high degree of dynamism, flexibility, modularity of application development process, ease of application interface extensibility, direct user participation, on-demand interface generation, and user centric interfaces. These benefits are succinctly summarized as follows:

a. Dynamism

The framework addresses the challenge posed by dynamic computing environment, which involves changes in set of consumers, providers and their preferences. CoMADE, automatically generates an interface to suit the state of a user or his/her preferences at a given time

b. Flexibility

Interface users, such as the consumers and providers of mobile grid services are provided with various means of interacting with CoMADE system. Users can specify example preferences and generate sample or specimen interfaces through the use of a proxy device. The introduction of a proxy device induces flexibility by providing users the opportunity to view a self-generated interface and can decide to request CoMADE for the delivery of such an interface.

c. Modularity of Application Development Process

Multi-target user interface development process of our framework involves design time creation of polymorphic artifacts, which are used during interface generation. Designers

or service developers can work on different polymorphic artifacts. A designer can enable a polymorphic artifact with an implementation module to achieve an alternative form of task execution different from another artifact handled by a counterpart designer.

d. Ease of Application Interface Extensibility

With CoMADE, an application interface can be easily extended by introducing one or more polymorphic artifact or task variants for addressing a newly defined user need or one or more user preferences. The newly introduced artifact(s) is/are included into the pool of variants from which variants are selected during an interface generation.

e. Direct User Participation

The selection of an artifact during interface generation for a requesting user is based on user preferences. CoMADE achieves direct user participation by allowing users to determine the nature of their application interfaces. A set of user-aware artifacts make up a user centric, multi-target user interface.

f. User Centric Interfaces

With the design time creation of artifacts using a predefined user model, and direct user participation for the selection of polymorphic artifacts using real-time user preference information, user interfaces are generated to suit a user. The evaluation result shows the capability of CoMADE in the generation of user centric, multi-target user interfaces.

e. On-Demand User Interface Generation

Our framework introduces a method of interface composition and a mediator component for the selection of interface artifacts for user interface generation. User interfaces are

not only achieved at design time as proposed by many existing methods, but "composed" or generated dynamically based on user preferences during the Load time.

Following the benefits of our framework presented above, we discuss a comparative analysis of three model-based design approaches to user interface design, namely, (A) multiple logical descriptions by Mori, et al. (2004a), (B) Intent-oriented by Bisignano, et al. (2006) and (C) our Polymorphic Logical Description model. Similar to the approaches proposed in (A) and (B), the PLD model achieves multi-device interface adaptation due the use of the J2ME platform as its target platform. The J2ME platform is cross-platform compatible. Our work and Mori, et al. (2004a) provide an implementation of a support toolkit or an authoring environment for user interface generation, namely, CoMADE and TERESA respectively. The implementation of a support toolkit for model based user interface design has the advantage of allowing designers to focus on more important aspects rather than being worried with implementation details. The implementation details are handled by the toolkit (Paterno, 2005).

PLD model through the load and runtime tiers provide support for run time adaptation. Similarly, the intent oriented approach (B) (Bisignano, et al. 2006) supports the adaptation of interface contents, such as media contents during the runtime but the capability is not supported in A (Mori, et al., 2004a).

PLD model offers unique advantages over the other two approaches. First, PLD uniquely supports direct user participation and on-demand user interface generation. PLD allows user involvement during interface generation and also, user interfaces can be generated automatically in response to real-time user preference information. These capabilities of PLD become highly relevant in a mobile computing environment where

service request information, user preferences and devices characteristics are not predetermined.

Second, our model achieves user centric interface with the aid of the CoMADE toolkit, which automatically generates an interface for individual users.

Third, the PLD model creates a flexible environment for end-user by the introduction of a user interface shopping cart for managing user requests, which are submitted through of a proxy device (less-constrained device). The use of proxy device for submitting user preference information and placing interface requests gives users the opportunity to view a self-generated interface, which is discarded or demanded for delivery to user mobile device. Moreover, our model makes provision for submission of user interface requests through the SMS technology. The other two approaches by Mori, et al. (2004a) and Bisignano, et al. (2006) do not provide a flexible environment for users. A flexible environment achieves universal usability due to the availability of compelling alternatives for different usage scenarios.

An important attribute of a software system is extensibility. Extensibility refers to the ability to provide support for new devices, languages, new interaction modalities with little or no cost overhead. Similar to TERESA by Mori, et al. (2004a), the CoMADE toolkit supports new target languages, device platforms and also an interface can be easily extended by introduction of one or more artifacts into a pool of polymorphic artifacts created by designers.

Table 5.1 gives a comparison between three model-based design models, namely, **(A)** multiple logical description by Mori, et al. (2004a), **(B)** Intent-oriented by Bisignano, et

al. (2006) and (C) Polymorphic Logical Description model proposed in this study (Ipadeola, et al. 2008b) using the eight criteria discussed above.

Table 5. 1: Comparative Analysis of Model-Based Approaches in User Interface Design

Evaluation Criteria	(A) Multiple Logical Description Model	(B) Intent-oriented Approach	(C) Polymorphic Logical Description
Multi-device adaptation	Yes	Yes	Yes
ToolKit implementation	Yes (TERESA)	No	Yes (CoMADE)
Run-time adaptation	No	Yes	Yes
Direct User Participation	No	Limited to Choice of Media Content	Yes
On-Demand Interface Generation	No	No	Yes
User Centric Interface	No	Partially	Yes
Flexibility	No	No	Yes
Ease of Extensibility	Yes	No	Yes

5.2 Conclusion

This work proposes a polymorphic logical description method for the development of multi-target user interfaces. The iPLD framework and its constituent PLD model are created to allow direct user participation in interface generation for universal usability of interfaces.

The result of our evaluation shows that our approach will have significant impacts in a mobile and grid environment. Evidently, achieving automatic interface generation in accordance to some specified requirements or preferences will enable universal accessibility to applications and services. The approach will save time and cost of designing interfaces for separate users.

Moreover, our approach will be of significant benefit to providers and consumers of mobile applications and services. It will enhance the experience of users of solutions in m-grid services, e-Commerce, e-Business, e-Learning, e-health and e-tourism, as they are able to vary the composition and usability of their interface at a given time.

5.3 Future Work

An important progression of this work will include achieving flexible user interface migration between federated devices interfaces for enabling task continuity and uninterrupted information communication. Additionally, this study aims to consider dynamic multiple interaction modalities such as the vocal interface in the nearest future.

BIBLIOGRAPHY

Aaron, M. (2003). Universal, ubiquitous, user-interface design for the disabled and elderly. Interactions. 10(2), 23-27.

Aaron, M. (2002). Dare We Define User-Interface Design?. Interactions, 9(5),19-24.

Adigun, M.O. (2006). Zululand University: on our way to wireless grid computing. Presentation at University of Syracuse, USA.

Annett, J. & Duncan, K. D. (1967). Task analysis and training design. Journal of Occupational Psychology. 41, 211-221.

Anil, S., Juan, Q., Sergiu M. D., Sushil J. L. & Monica N. N. Sycophant (2007): An API for Research in Context-Aware User Interfaces. In the Proceedings of the 2nd International Conference on Software Engineering Advances (ICSEA). Cap Esterel. 83-83

Banavar, G., Bergman, L., Cardone, R., Chevalier, V., Gaeremynck, Y., Giraud,F., Halverson, C., Hirose, S., Hori, M., Kitayama, F., Kondoh, G., Kundu, A., Ono, K.,

Schade, A., Soroker, D. & Winz, K. (2004). An Authoring Technology for Multi-device Web Applications, IEEE Pervasive Computing. 3(3), 83 – 93.

Barbara B. (2007). Designing Mobile User Experience. John Wiley & Sons, ISBN: 978-0-470-03361-6.

Bisignano, M., Di-Modica, G. & Tomarchio, O. (2006). An Intent-oriented Approach for Multi-Device User Interface Design. In Proceedings of the 20th International Conference on Advanced Information Networking and Application., AINA(2),186 – 194.

Bitpipe, (2007). Available Online: http://www.bitpipe.com/tlist/Web-Applications-Software.html. (Last accessed on 15 June 2008).

Braun, E., Hart, A. & Muhlhauser, M. (2004). Authoring for Multi-Device Interfaces. In the Adjunct Proceedings of 8th ERCIM Workshop. 186 – 194.

Brewster, S., Leplatre, G., & Crease, M., (1998). Using Non-speech Sounds in Mobile Computing Devices. In Proceedings of the First Workshop on Human Computer Interaction of Mobile Devices. Glasgow. 5(3), 224-259.

Burns, J., Gregory, R.M. (2001).A Framework for Effective User Interface Design for Web-Based Electronic Commerce Applications. International Journal of an Emerging Transdiscipline. 4, 67-75.

Correani, F., Mori, G. & Paternò, F. (2004). Supporting Flexible Development of Multi-Device Interfaces. In the Proceedings of EHCI-DSVIS, Hamburg.

Constantine, S. (2000). User Interfaces for All: Concepts, Methods and Tools. Lawrence Erlbaum Associates, Inc., Mahwah, NJ. ISBN: 978-0-8058-2967-9.

Cristina C., Paterno, F. & Santoro, C. (2004). Methods and Tools for Designing and Developing Usable Multi-Platform Interactive Applications. PsychNology, 2 (1), 123-139.

Culler D., Estrin D., & Srivastava M. (2004). Guest Editors' Introduction: Overview of Sensor Networks. IEEE Computer. 37(8), 41-49.

David, T., Joelle, C., & Gaelle, C. (2001). A Unifying Reference Framework for the Development of Plastic User Interfaces. In Proceedings of the 2001 Engineering of Human—Computer Interaction Conference (EHCI'2001). 173-192.

David E., Arouna W., Feng T., Hugh C. (2005). Experiences with Writing Grid Clients for Mobile Devices. In Proceedings of 1st International ELeGI Conference, Vico Equense - Napoli(Italy).

Eisenstein, J., Vanderdonckt, J. & Puerta, A. (2001). Applying Model-Based Techniques to the Development of UIs for Mobile Computers. In Proceedings of the ACM Conference on Intelligent User Interfaces. 69–76.

Forman, H.G. & John Z. (1994). The Challenges of Mobile Computing. IEEE Computer. 27(4), 38-47.

Genco, A., Sorce, S., Reina, G., & Santoro, G. (2006). An agent-based service network for personal mobile devices. IEEE Pervasive Computing. 5(2), 54 – 61.

Germanakos P., Samaras G., Mourlas C. & Christodoulou E. (2006). Innovative Personalization Issues for Providing User-Centric mGovernment Services. In Proceedings of the Second European Conference on Mobile Government. 124-133.

Gibbons P., Karp B., Ke Y., Nath S., & Seshan, S. (2003). IrisNet: An Architecture for a Worldwide Sensor Web. IEEE Pervasive Computing. 2(4), 22-33.

Graef, G.. & Gaedke, M. (2000).Construction of Adaptive Web-Applications from Reusable Components. In 1st International Conference on Electronic Commerce and Web Technologies (EC-Web). 1-12.

Grundy, J.C. & Jin, W. (2002). Experiences developing a thin-client, multi-device travel planning application. In Proceedings of 2002 New Zealand Conference on Computer-Human Interaction, July 12-13, Hamilton, New Zealand.

Guan, T., Zaluska, E., & Roure, D. (2005). A Grid Service Infrastructure for Mobile Devices. In 1st Semantic Knowledge and Grid Conference.

Herrero, C. & Lamadon. J.L. (2004). A Device Independent User Interface: a television program guide case study. Research Seminar on Digital Media. Available online:

http://www.tml.tkk.fi/Studies/T-111.590/2004/. (Last accessed on 30 June, 2008).

Huang, C.M. & Chao, Y.C., (2001). Universal WWW Access for Heterogeneous Client Devices. In the proceedings of the 27th Euromicro Conference 2001. 315.

Ipadeola, A.O., Iyilade J.S, Adigun, M.O. & Xulu S.S. (2008). A Multi-target User interface Design for Pervasive Grid Environment. International Symposium on Pervasive Grid (PerGrid08). Brazil. (Accepted forPublication).

Ipadeola, A.O., Olugbara O.O, Xulu S.S & Adigun, M.O. (2008). Polymorphic Logical Description for Automatic generation of User Centric, Multi-Device Interfaces. The third International Conference on Pervasive Computing. Alexandra Egypt. (Accepted for Publication).

Ipadeola, A.O., Olugbara O.O, Adigun, M.O. & Xulu S.S. (2008). A System for Dynamically Generating User Centric Interfaces for Mobile Applications and Services. International Workshop on Modelling Mobile Applications and Services. Barcelonia, Spain. (Accepted for Publication).

Ipadeola, A.O., Olugbara O.O, Adigun, M.O. & Xulu S.S. (2008). Automatic Generation of User Centric Interfaces for J2ME Enabled Handheld Devices. The Southern African telecommunications Networks and Applications Conference (SATNAC).Eastern Cape, SouthAfrica. (Accepted for Publication).

Isaiadis, S. and Getov, V. (2005). A lightweight platform for integration of mobile devices into pervasive grids. High Performance Computing and Communcations. In

Proceedings. Lecture notes in computer science (3726). Springer, Berlin, Germany, 1058-1063. ISBN 3540290311.

Jacob, E., Jean, V. & Angel, P. (2001). Applying Model-Based Techniques to the Development of UIs for Mobile Computers. In Proceedings of the ACM Conference on Intelligent User Interfaces (IUI'2001). 69–76.

James, L. (2002). Damask A Tool for Early-Stage Design and Prototyping of Cross-Device User Interfaces. In Conference Supplement of UIST 2003: ACM Symposium on User Interface Software and Technology. 13–16.

Jank, M. & Pospischil, G. (2005). Device Independent Mobile Multimodal User Interfaces with the MONA Multimodal Presentation Server. In Proceedings of Eurescom Summit. 3(7), 89-93.

John D., Jacob P.U. & Stephen J.B. (1993). Improving Application Development Productivity by Using ITS. International Journal of Man-Machine Studies. 39(1), 113-146.

John, H. & Seely, J. Service grid: The Missing Link in Web Services. Available Online: www.johnhagel.com/paper_servicegrid.pdf (Last accessed on 22 June, 2008).

Kai, R. (2005). A Transformational Approach to Multi-Device interface. In proceedings of human Factor in computer System, Portand Oregon, USA.

Kansal, A., Goraczko, M., & Feng, Z. (2007). Building a sensor network of mobile phones in Information Processing in sensor Networks. In Information Pro-cessing in sensor Networks (IPSN).

Keith, E.W, & Grinte, R.E. (2001). At Home with Ubiquitous Computing: Seven Challenges. In the Proceedings of the 3rd International Conference Atlanta, Georgia, USA. 2201/2001.

Koudouna V., & Omar L. (1996). Mobile Computing: Past, Present and Future" Internation Journal of Surprise AvailableOnline: http://www.doc.ic.ac.uk/ ~nd/surprise_96/ journal/vol4/vk5/report.html. (Last accessed 8th July).

Lonczewski, F, & Shreifer, S. (1996). The FUSE System: an Integrated User Interface Design Environment, In Proceedings of the working conference on Advanced visual interfaces. 37-56.

Mori, G., Paterno, F. & Santoro, C. (2004a). Design and Development of Multi-device User Interface through Multiple Logical Descriptions. In Proceedings on Software Engineering. 30(8), 507-520.

Mori, G., Paterno, F. & Santoro C. (2004b). CTTE: Support for Developing and Analyzing Task Models for Interactive System Design. IEEE Transactions on Software Engineering. 28(8), 507-520.

Myers, B., Hudson, S., Pausch, R. (2000). Past, Present, Future of User Interface Tools. Transactions on Computer-Human Interaction. ACM 7 (1),3–28.

Nasseam, E. (2003). A Web Services Strategy for Mobile Phones. Available Online: http://webservices.xml.com/pub/a/ws/2003/08/19/mobile.html. (Last accessed on 4 June 2008)

Nielsen J. (2000). Why You Only Need to Test with 5 Users. Available Online: http://www.useit.com/alertbox/20000319.html. (Last accessed on 4 June 2008)

Niklfeld, G., Anegg, H., Gassner, A., Jank, M., Pospischil, G., Pucher, M., Schatz, R., Simon, R. & Wegscheider, F. (2005). Device Independent Mobile Multimodal User Interfaces with the MONA Multimodal Presentation Server. Eurescom Summit.

Olsen, D., Jefferies, S., Nielsen, T., Moyes, W. & Fredrickson, P. (2000). Cross-modal Interaction using XWeb. In Proceedings of the 13th Annual Symposium on User Interface Software and Technology CA, USA.

Paterno, F. (2005). Model-Based Tools for Pervasive Usability. Interacting with Computers. 17(3). 219-315

Paterno, F. (2005). Multimodality and Multi-device interfaces. In W3C Workshop on Multimodal. Interaction.

Paterno F., Santoro, C. (2002). One model, many interfaces. In Proceedings of the 4th International Conference on Computer-Aided Design of User Interfaces (CADUI'2002). 143-154.

Puerta, A.R. & Szkeley, R. (1994). Model-based interface development. In Conference companion on Human factors in computing systems. 14(4), 40-47.

Richter, K. (2005). A Transformation Strategy for Multi-device Menus and Toolbars, CHI '05 Extended Abstracts on Human Factors in Computing Systems. 1741-17

Rodrigo, O.(2006). Mobile Access to Web Systems Using a Multi-Device Interface Design. In Proceedings of 2006 World congress in Computer Science, Computer Engineering and Applied Computing. 332-334.

Savidis, A., Alex, P., Demosthenes, A. & Constantine, S. (1997). Designing user-adapted interfaces: the unified design method for transformable interactions. Symposium on Designing Interactive Systems. 323-334.

Seffah, A., Forbig, P. & Javahery, H. (2004). Multi-devices Multiple User Interfaces: Development Models and Research Opportunities. The Journal of Systems and Software.73. 287-300.

Schlungbaum, E. (1997). Individual User Interfaces and Model Based User Interface Software Tools. In Proceedings of International Conference on Intelligent User Interfaces.

Szekely, P. (1996). Retrospective and Challenges for Model-Based Interface Development. In Proceedings of the 2nd International Workshop on Computer-Aided Design of User Interfaces.

Szekely, P. (1995). Declarative Interface Models for User Interface Construction Tools: The Mastermind Approach. In Engineering for Human-Computer Interaction. 120-150.

Tao, G., Zaluska, E. & David R. A Grid Service Infrastructure for mobile Devices. In Proceedings of 1st Semantic Knowledge and Grid conference. Beijing, China.

Vanderdonckt J.M, & Bodart F., (1993) Encapsulating knowledge for intelligent automatic interaction objects selection. In Proceedings of the SIGCHI conference on Human factors in computing systems, 424—429.

Vincent, S. (2002). Pervasive Computing Goes to Work: Interfacing to the Enterprise. IEEE Pervasive Computing, 1(3), 6-12.

Weicha, C., William, B., Stephen, B., John, G. & Sharon Greene. (1990). ITS: A tool for rapidly developing interactive applications. ACM Transactions on Information Systems (TOIS). 8(3). 204 - 236

Yuan M, & Long, Ju. (2004). Securing Wireless J2ME, Security Challenges and Solutions for Mobile Commerce Applications. Available online: http://www.ibm.com/developerworks/wireless/library/wi-secj2me.html. (Last accessed on 20 July 2006)

Zhigang H., & Hanqing, L. (2006). Web Browsing on Small-Screen Devices: A Multiclient Collaborative Approach. IEEE Pervasive Computing. 5(2), 78-84.

APPENDIX A

Questionnaire

This questionnaire, which is sub-divided into two parts, will help in the evaluation of a polymorphic task modeler which was implemented for the design of multi-target user interfaces.

Kindly provide answers to the following questions:

1. How often do you use the computer? (Circle One)

   All day Once/day once/week Once/month Less Frequently

2. How long have you been using the computer?

   A. 1-2 years B. 2 – 4 years C. Greater Than 4years

3. Do you have a personal computer? A.Yes B.No

4. Can you work with Microsoft Office Package? A.Yes B.No

5. Have you worked with any of the following?

   Corel Draw, Photoshop, Page maker a. Yes b. No

6. Have you undergone any computer training course?

   a. Yes b. No If yes Mention the course.....

7. How many years of experience do you have in user interface design?

   (A) Less than 2 years (B) 2- 3 years (C) Greater Than 3 years

8. How do you best describe yourself in terms of user interface design?

(A) Professional Designer (B) Intermediate Designer (C) Novice Designer

9. Which of the following best describes you?

(A) Full Time Programmer (B) Intermittent Programmer (C)A Beginner

10. Name the Integrated Development Environments (IDE) that you have worked with?

..... (e.g) Grasp, Net Beans, JBUILDER, Eclipse IDE etc

11. I found the system effective in the creation of polymorphic task models

Strongly Disagree	Disagree	Undecided	Agree	Strongly Agree

12. I found the system effective in the creation of polymorphic abstract models

Strongly Disagree	Disagree	Undecided	Agree	Strongly Agree

13. I found the system effective in the creation of concrete model?

Strongly Disagree	Disagree	Undecided	Agree	Strongly Agree

14. How effective is the system in generating the final user interface?

Strongly Disagree	Disagree	Undecided	Agree	Strongly Agree

15. How satisfactory is the use of the modeler? Is it user friendly and easy to use?

Very Hard	Hard	Moderate	Easy	Very Easy

16. How fast is the system in user interface modeling and design?

Very Slow	Slow	moderate	Fast	Very fast

17. The *initial* user interface generated met the preferences I specified.

Strongly Disagree	Disagree	Undecided	Agree	Strongly Agree

18. Whenever I specify new preferences on the *initially* generated user interface, the interface adapts quite well to the new preferences.

Strongly Disagree	Disagree	Undecided	Agree	Strongly Agree

APPENDIX B

The CoMADE Toolkit

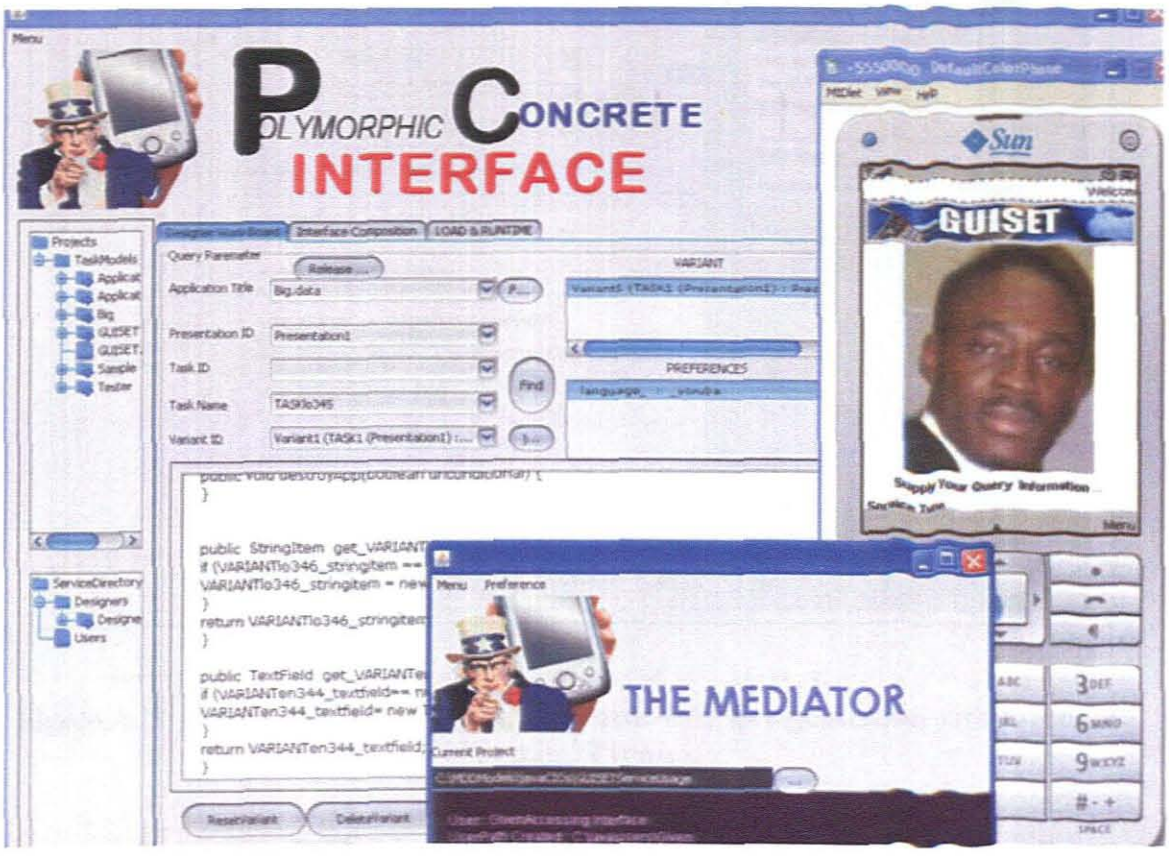


Figure A. 1: CoMADE Toolkit during Interface generation with an Emulator Program Type1

Figure A. 2: CoMADE Toolkit during Automatic Interface generation with Emulator Program. Type1.

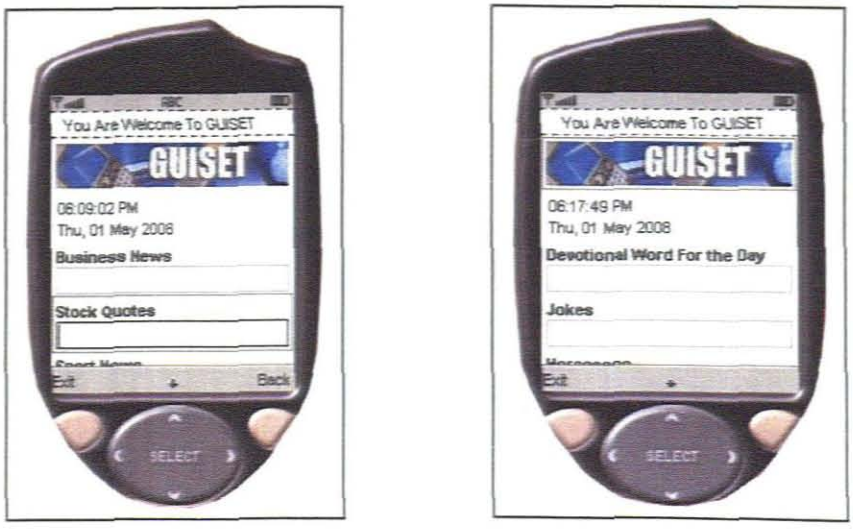


Figure A. 3: Interface Variants Generated by CoMADE

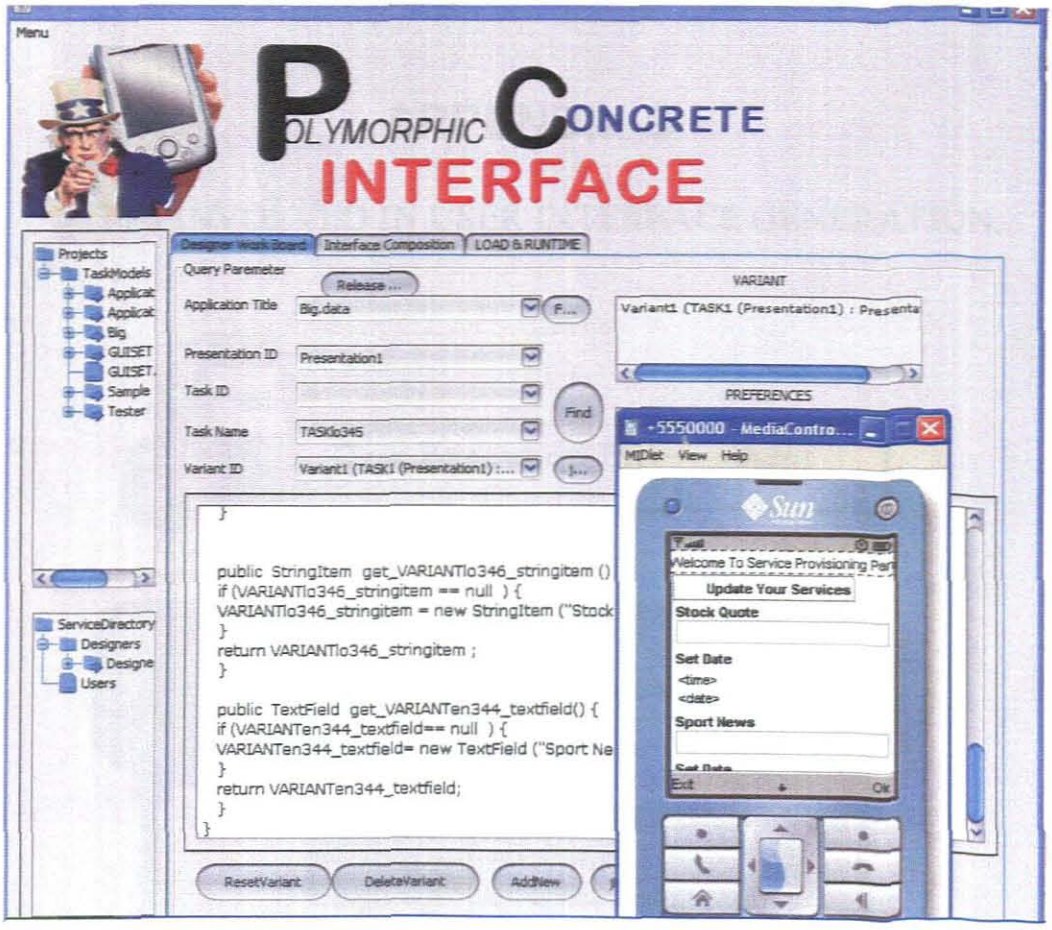


Figure A. 4: CoMADE Toolkit during Automatic Interface generation with Emulator Program. Type3.

APPENDIX C

STEPS INVOLVED IN USER INTERFACE GENERATION.

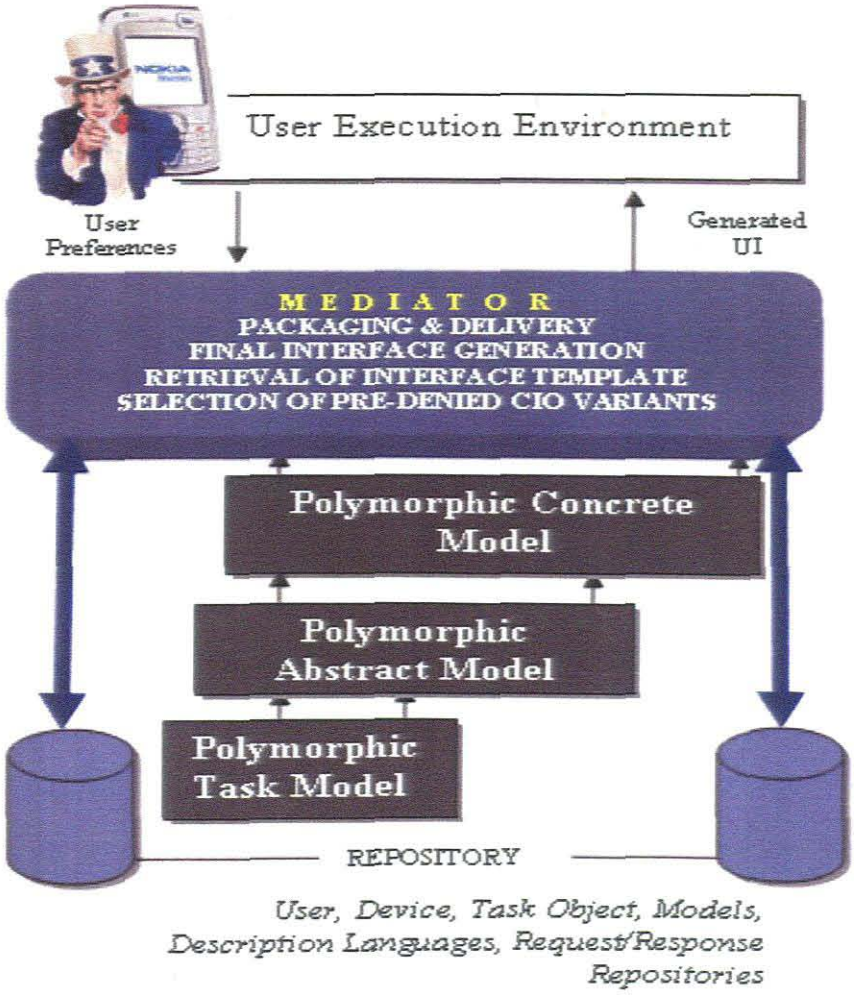


Figure A. 5: Steps Involved in User interface Generation