MODEL BASED APPROACH FOR CONTEXT AWARE AND ADAPTIVE USER INTERFACE GENERATION

REENA G. HANUMANSETTY

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Science and Applications

Dr. Denis Gracanin, Chairman
Dr. Mohamed Y. Eltoweissy, Committee member
Dr. Shawn A. Bohner, Committee member

July 20, 2004
Falls Church, Virginia

Keywords: Context awareness, Adaptive User Interface (UI), model based UI design, XML based UI languages, Concurrent Task Tree (CTT)

MODEL BASED APPROACH FOR CONTEXT AWARE AND ADAPTIVE USER INTERFACE GENERATION

REENA G. HANUMANSETTY

ABSTRACT

User interface design and development for ubiquitous software applications is challenged by the presence of varying contexts. Context comprises of user's computing platform, the environment in which the user is interacting with the application and user characteristics which comprise of user's behavior during interaction and user preferences for interface display and interaction. We present a framework for adaptive user interface generation where adaptation occurs when context changes. This framework introduces three new concepts. First, formalization for representing context is introduced. Our design of context specification is unique since it reflects the association of context with level and nature of user interface adaptation. Secondly, user interface generation life cycle is studied and we define a context model on top of task model to introduce the contextual conditions into user interface generation process. Using the context model, user interface designer can specify contextual requirements and its effect on the user interface. Third, context aware adaptation of user interfaces is achieved by mapping context specifications to various levels of user interface generation life cycle. We designed a specification language called rule specification using which the user interface designer can specify the mapping. With the new design of context representation, context model, and rule specification, we demonstrate how changes in contexts adapts task model which in turn adapts the user interface.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my thesis advisor, Dr. Denis Gracanin, for his valuable guidance, encouragement, and for taking time off his busy schedule to provide invaluable input towards completing my Master's thesis. I would also like to thank him for the laboratory resources. I am grateful to Dr. Mohamed Eltoweissy who has also guided me and helped me improve my researching skills. I would also like to thank Dr. Shawn Bohner for his valuable input and guidance while working towards my thesis.

I extend my gratitude to all the researchers involved in CTT, and USIXML mailing lists who shared their views and provided their insight into my views. I would like to acknowledge and thank Ph.D. students Xiaoyu Zhang, and Boby George for their input while working towards my thesis. I appreciate all support provided by faculty, staff and my colleagues in the Department of Computer Science, Virginia Tech in National Capital Region.

I would also like to thank my parents and my brother's family for all their support, encouragement, care, and good wishes to help me complete my thesis. Thank you God for everything. Thank you all for making me a better person.

DEDICATIONS

To The Almighty God
My Mom Jyothi, Dad Mallikarjuna Rao,
Advisor Dr. Denis Gracanin,
Brother Naveen and Sister-in-law Ambica

Without all your encouragement, care, support, understanding, and good wishes, this thesis and my Master's degree would not have been possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

In this chapter, we present the motivation for this thesis, research issues, challenges, the approach we have chosen to address these issues, and thesis contributions. We conclude this chapter with the structure of this report.

1.1   MOTIVATION

With increasing number of variety of devices and their use in ubiquitous environments by various types of users, adaptation of user interfaces has become a necessity rather than a facility. Existing systems lack the ability to satisfy the heterogeneous needs of many users. These needs also vary according to the context, where context comprises of environmental conditions, the characteristics of device used to interact with the application, and the user characteristics. There is a need for techniques that can help the UI designer and developer to deal with myriad of contextual situations. Also, user should be provided with the facility to have an adaptive interface that adapts to changing needs of the user.

A sample application here explains the need for adaptive user interface generation. Consider an application being developed to allow access to patient record information. There are various kinds of users who accesses patient records. These users can be broadly classified into two types [2]: Individual users and organizational users. Individual users include patients, physicians, dentists, dieticians, Lab technologists, occupational therapists, optometrists, pharmacists, healthcare administrators, and organizational users include but are not limited to healthcare providers, provider networks, health plans, donor banks, and ambulatory surgery centers. Users such as patients have certain limitations and use of system for people with disabilities and/or illness requires the interface to be friendly and adapt to their static and dynamic behaviors. Users such as physicians often accesses patient records using a desktop, a Personal Digital Assistant (PDA) or a Tablet and require that the interface and interacting techniques change to their current activity and location. The generated interface not only depends on user's static characteristics such as the role, permissions, and pre-defined preferences but also on dynamic characteristics including mouse movement, user's activity, environment conditions and user's current location. Hence, each user varies in their needs of user interface and each time the application may be accessed from different devices or computing platforms. The user interface facets including content, structure, navigation, presentation and style have to be modified based on the context. Technological support is necessary for design, development and generation of user interface to each computing platform, each type of user and each type of environment.

1.2   RESEARCH ISSUES AND CHALLENGES

Adaptation of the interface occurs when there is change in the "context". A number of contextual parameters, location and computing platform to name a few, are associated with context. There are a myriad of systems that addressed the problem of adapting user interface to different platforms but very few that addressed the challenge of adapting the

interface to location. Requirements of a system that adapts its user interface to different platforms are different from the requirements of a system that adapt to changes in location. With varying requirements and various origins of context, the following research issues and challenges arise:

1. *Build a user interface that dynamically adapts to the context*
2. *Representation and handling of contexts that differ in origin and nature*
3. *A unified framework for context aware adaptive user interfaces that can address varying requirements related to various contextual parameters or seamless integration of context handling and user interface generation.*

Generation of user interface using model based approach is a combination of manual and automatic process. While complete automation is neither essential nor feasible, automation needs to exist with manual user interface design process in order to generate the interface at run time and generation of user interface at run time is essential to meet the changing contexts which change the needs of user and the interface. Hence automation of the user interface generation, to the maximum extent possible, is one of the key research issues.

While there are number of challenges to user interface development, we concentrate on those challenges that are pertaining to adaptive user interfaces. Since adaptation occurs when there is change on 'context', handling adaptation of each type of contextual parameter can be considered as a challenge. By having an adaptive user interface development, the user interface is developed only once, however this can be used in multiple contexts since the user interface adapts to the context. Following is a list of challenges pertaining to adaptive user interfaces:
- Ability to handle both input and output using multiple mechanisms. This requirement arises not only because of limited interaction capabilities of certain devices but also because of environmental conditions like user's current task. For example, limitations of mobile phones which do not have a keyboard require the application running on it to accommodate for alternate input mechanisms that does not require keyboard input. User might be involved in a task, such as driving, in which case the user may wish to interact with the application using speech as input and get a response back in the form of voice. Hence, a user interface that has this ability decides on the interaction mechanism depending upon the contextual parameters like device characteristics and user's tasks.
- Support for multiple metaphors: Choice of metaphors is made not only based on the screen size of the display device but also based on user preferences.
- Support for multiple platforms: User interface development should involve support of "write once, deploy anywhere" property called platform independence which support different platforms.
- Support for automatic adaptation of interfaces
  o To multiple platforms: User interfaces should be platform aware which implies that the interface and interaction should change to platform characteristics such as display device resolution, and available interaction mechanisms.

2

- o To users behavior where users vary in their capabilities of interaction
- o To individual user's preferences
- o To user tasks and current needs and apart from user's privileges
- o To environmental conditions such as users located near by, noise level, lighting conditions, and available network bandwidth.

1.3   APPROACH

The approach we have used to achieve the goal of context aware adaptation of user interface is to build upon model based design techniques and context frameworks and integrate them to transit from context specifications to user interface generation. Use of task model for user interface modeling has been well accepted in the process of interaction design. Hence we build an abstraction above the task model called the context model, which is based on the Concurrent Task Tree (CTT) notation developed by Fabio Paterno [66]. Our context specification is based on W3C standard, Composite Capabilities / Preference Profile (CC/PP), for expressing device capabilities and user preferences.

1.4   THESIS CONTRIBUTIONS

Contributions of the thesis include introduction of new techniques to represent context and context model.

1. *A unique way to specify and handle context when used in domain of user interfaces.*
    Primary context (static and dynamic) specification language and action specification language introduced in this thesis are intended for use by broad range of applications that require its interface to be adaptive. This specification provides a means for the user interface (UI) designer to specify the *cause* of change in user interfaces. It also provides a basis for run time adaptation of user interface.
2. *Techniques to specify a context model on top of a task model*
    User interface designer is provided with the facility to model context sensitive user interfaces with a context model. Context model introduced in this thesis provides increased flexibility to the UI designer to specify the *effects and results* of contextual conditions on the task specification of user interface. It bridges the gap between context specification and task model.
3. *Techniques for rule specification*
    User interface designer is provided with the facility to specify rules using rule specification language. This also provides the facility to specify effects and results of contextual conditions. Here, conditions may be specified at various levels of user interface generation life cycle including but not limited to dialog model that results in , and presentation model.
4. *Study of existing context aware frameworks, similarities and differences*
    Frameworks that abstract sensory context from the applications have been studied and presented. Such a framework forms useful part of applications that require

3

adaptation to sensory contexts where sensory contexts are those contextual information obtained from sensors. Examples include, location, ambient and aural condition of the environment.

5. *A comparative analysis of model based user interface development environments*
Analysis of models including task, domain, user, dialog, and presentation model explored in various projects and tools used to support these models formed the foundation for defining the context model contributed in this thesis.

6. *Extensive study of XML based user interface description languages*
User interface description languages are essential part of adaptive user interfaces since various general eXtensible Markup Language (XML) based UI description languages can be translated to specific description languages such as Hyper text Markup Language (HTML), Java, Wireless Markup Language (WML), and VoiceXML.

7. *A basis for future work*
Ideas presented in this thesis and the manner in which this work can be used in other on-going research projects at Virginia Tech, provide enough starting points and motivation to utilize and augment this thesis work.

1.5   SCOPE

From the perspective of concepts introduced in this thesis, user interface generation life cycle starts with context specification, and continues with context model, task model, dialog model, and presentation model. Changes in contexts change the contextual requirements that generate the user interface. In order to achieve adaptation of user interface, changes in contexts described in context specification propagate to any level of the UI generation life cycle. This thesis concentrates on the context and task level. Contextual conditions that require change in presentation, style and layout needs to propagate to models below task model. Although theoretical concepts and existing work related to transition form task model to UI generation have been described in this thesis, our contributions concentrate on evolving task model from context specification and context model.

1.6   DEFINITIONS

 "**Context** is any information used to characterize situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves." [8, 9]

"**Context awareness** is the facility to establish context." [8, 9]

Context aware user interface is any user interface that changes when context changes. Context aware user interfaces are either adaptive or adaptable user interfaces. Adaptive user interface is one in which system initiates the context change. Adaptable user interface is one in which the user initiates the change in context.

Context sensitive user interface in this document is interchangeably used with context aware user interface.

"**Task** is an activity that should be performed in order to reach a goal. A goal is either a desired modification to a state or an enquiry to obtain information on the current state." [106]. "A **model** captures some facets of the problem and translates them into specifications" [103]. "A **task model** describes tasks that users need to perform in order to reach a goal when interacting with a computer based system." [65]. [106] defines task model as a description of an interactive task to be performed by the user of an application through the application's user interface. There is currently no agreement on how a task can be adequately represented and hence definition of task model is varying. **ConcurTaskTree** (CTT), short form for concurrent task tree, is a type of task model, details of which are given in chapter 2.

1.7   REPORT STRUCTURE

Chapter 2 provides review of literature on context awareness in general, its frameworks and applications. Chapter 2 also presents a review on adaptive user interfaces, model based user interface development technologies, and XML based user interface description languages. Chapter 3 presents the new framework, design and implementation of our prototype. To prove the concepts introduced in chapter 3, the next chapter illustrates the working with sample application. We conclude with Chapter 5 in which we discuss future directions to enhance this work and open issues relevant to this thesis topic. Appendix shows the details of the schemas designed for context model, context specification, rule specification and files related to example application of chapter 4.

CHAPTER 2: LITERATURE REVIEW

In this chapter we present a literature review of context awareness, and adaptive user interfaces. In the survey of context awareness, we start with definition of context, the various frameworks that use and process context, and the applications that have implemented context aware applications. In the review of adaptive user interfaces, we present the quality metrics for user interfaces, architectures and toolkits that adapted user interfaces to context, and projects that illustrate adaptive user interfaces. We continue with a study and analysis of various model based user interface development environments and XML- based UI description languages.

2.1 INTRODUCTION TO CONTEXT

2.1.1 Defining Context

While context has been defined in numerous ways, we present here two frequently used definitions. Dey and Abowd [8, 9] define context, context awareness and context aware applications as:

Definition: "Context is any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves. A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task. Context awareness is the facility to establish context. Context aware applications adapt according to location of use, collection of nearby people, hosts and accessible devices, and their changes over time. The application examines the computing environment and reacts to changes"

Chen and Kotz [31] define context by making a distinction between what is relevant and what is critical.
Definition: "Context is a set of environmental states and settings that either determines an application's behavior or in which an application event occurs and is interesting to the other user."
They define the former situation as a critical case called *active context* and the later one as a relevant one naming it as a *passive context*.

2.1.2 Types of contextual elements

Gross and Specht [29] list 4 dimensions of context: location, identity (e.g., user's interest, preferences, knowledge, and activity logs), time (e.g., working hours), environment or activity or task. Schilit, Adams and Want [30] states three important aspects of context: "Where are you? Who are you with? And what resources are near you?" In general there were several applications that used context and the various parameters observed from these existing applications include location, time, lighting, noise level, network connectivity, communication costs, physical activity, user's current task or activity, role or identity, need for privacy, social situation, history information, device type.

David Thevinin [45] categorizes contextual elements into users, platform and environment where users are the one accessing context aware application on a platform in an environment which might comprise of other users, or user's current tasks. A combination of user, platform and environment is termed as '*context of use*'.

2.1.3   *Classification of context aware applications*

Pascoe [34] presents four core features of context awareness: *Contextual sensing*, as the name suggests, is the ability to sense context information and present it to the user. *Contextual adaptation* is the ability to execute or modify a service automatically at run time based on the context. The ability to discover and use resources and services related to the current context is featured as *context resource discovery*. The fourth feature *contextual augmentation* is the ability to supplement digital data with the user's context.

Schilit [30] classifies context aware applications based on two dimensions: the operation of the task and the way in which task is executed. Each of the two dimensions takes two values. The operation may be to retrieve information or to execute a command. The second dimension may be either automatic or manual. Applications that retrieve information for the use manually based on available context are classified as *proximate selection applications*. Applications that retrieve information automatically based on context are called automatic contextual reconfiguration. Applications that execute operations manually are called *contextual command applications*. Applications that execute commands automatically for the user based on the context are called *context triggered actions*.

Operation\Automation	Automatic	Manual
Retrieve	Automatic contextual reconfiguration	Proximate selection applications
Execute	Context triggered actions	Contextual command applications

Table 1: Classification of context aware applications by Schilit

Contextual Augmentation feature is missing in Schilit's classification and contextual commands feature is missing in Pascoe's taxonomy. Dey and Abowd [9] combines the ideas of the above classifications and categorizes the features that context aware applications may provide into three classes: *presentation of information and services to user, automatic executions of a service, tagging of context to information for later retrieval.* They classify context types into 4 types: Activity, Identity, location and time.

Rhodes and Maes [32] classifies the context aware applications into reactive and proactive applications. Example of a proactive context aware application is Just-In-Time information retrieval agent (JITIR). JITIR agents are class of software agents that proactively present information to user's based on user's current context without requiring any action on part of user.

Brown et. al[12] classifies context aware applications into 6 types: *Proactive triggering, Streamlining Interaction, Memory for past events, Reminders for future contexts, Optimizing pattern behavior, Sharing experiences.*

2.1.4   Representation of context

Chen and Kotz [31] present the use of data structures to represent context. Context is represented as a set of parameters / attributes. Brown et. al [12] represents context in name/value pairs. The application sets the rules for determining whether the two values match. He also points that attribute values like "none", "any" and "not working" are needed to get all desired results. Tagged encoding is used in "Stick-e" note application [27] where contexts are modeled as tags and corresponding fields. The action of the <body> tag are triggered when constraints in <require> tag are met. This evolved into ConteXtML for exchanging information between user and server.

Composite Capabilities / Preference Profile (CC/PP) is a W3C standard for expressing device capabilities and preference profile. User with specific preferences and needs can specify these requirements using CC/PP. Creators of web devices and user agents can easily define precise profiles for their products. Web servers or proxies can use these profiles to adapt by tuning content selection or transformation. CC/PP also has the capability to trigger content transformation allowing a single source to be adapted to range of devices and user agents. [41]

2.2   FRAMEWORKS   AND   ARCHITECTURES   THAT   SUPPORT   CONTEXT-AWARE APPLICATIONS

We present in this section, existing frameworks that support development of context aware applications. We classify these frameworks based on the features of context aware applications that these frameworks support.

2.2.1.   Java Context Aware Framework

Java Context Aware Framework (JCAF) [28, 20] is the first of the kind to provide a Java based application framework. JCAF was developed to aid development of domain specific context aware applications. One of the motivations of JCAF is to have Java API for context awareness in much the same way JDBC is for databases and JMS is for messaging services. It is relatively new with compact and small set of interfaces defined in the API. Efforts are being put up to have more applications developed using this framework and further enhance it.

Architecture: JCAF is a *distributed, loosely-coupled*, service-oriented, event based, and secure infrastructure. Components of the JCAF framework include context service, access control, remote entity listener, context client, context monitor. The architecture is based on distributed model view controller and its design principle is based on semantic free modeling abstractions.

Context Client Tier   Context Service Tier
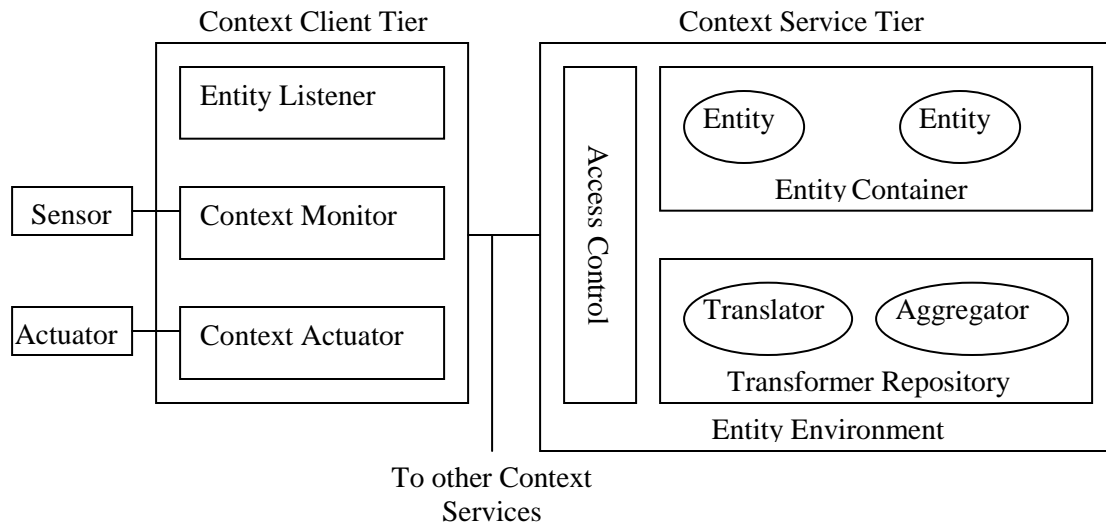
Figure 2.1: Run time architecture of Java Context Aware Framework [31]

Figure 2.1 shows the run time architecture of JCAF. JCAF infrastructure is a collection of context services connected in a peer-to-peer fashion where each context service is responsible for handling specific context information. It consists of a network of context services where each service can query the other service. Access to context service is through an access control component. Context clients can access context information by a request-response mechanism or by subscribing to an *entity listener*. [5, 6] Entities residing in Context Service are notified of any context events and the can share the context information with other entities. JCAF event mechanism is used for notifying the entities as well as for triggering actions.

Examples of applications developed using JCAF and the potential candidates for choosing JCAF. JCAF was intended to be for use in research and teaching. JCAF has been used in three of the projects [16, 19, 20]: *Proximity based user authentication, the context aware hospital bed, the AWARE framework.*

2.2.2.   Context Toolkit
The main objective behind development of context toolkit is to separate the context acquisition (process of acquiring context information) from the way it is used and delivered. Dey et. al [8] uses object oriented approach and introduced three abstractions: *widgets, servers and interpreters.* Context *widgets* are software components that were introduced in order to treat context as a form of user input, to abstract the way the context is sensed, and to provide easy access to context separating the concerns and provide facility for reuse. *Interpreter* is responsible for interpreting the context by transforming the raw context data and interpreting multiple contextual data. Context Servers are associated with each entity and all the contextual information related to that entity is collected by the corresponding context. The context server can be seen as a compounded widget that acts as a proxy to the application and is responsible for subscribing to every widget of interest. Context servers have attributes and callbacks, it can be subscribed to

9

and polled and its history can be retrieved [9]. Figure 2.2 shows the architecture of context toolkit.

To summarize, the services of context toolkit include abstraction of sensor information and context data through interpreters, access to context data through network API, sharing of context data through distributed infrastructure, storage of context data and basic access control for privacy protection.
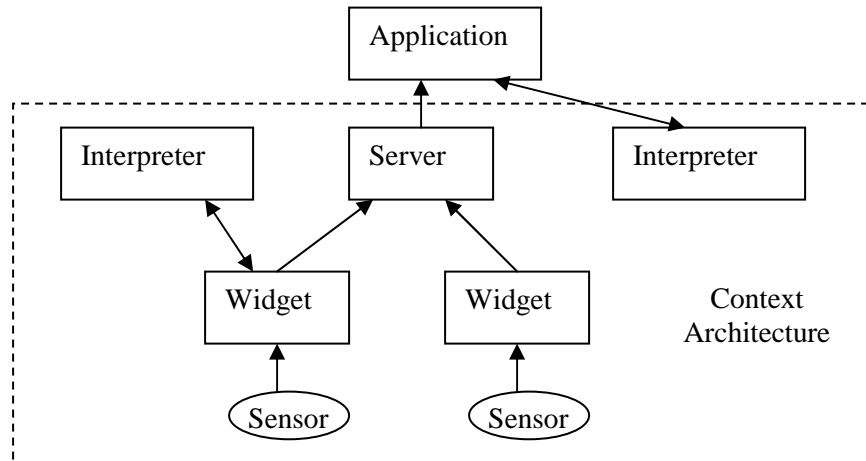
Figure 2.2: Architecture of Context Toolkit [10]

Examples: A number of applications have been developed using the Context Toolkit. One of the applications related to providing an interface to the user is an information display that shows the user standing in front of it, a URL related to the research group that they are in. A Dummbo application implemented using Context Toolkit uses a reader mounted to the whiteboard called Dummbo; when two users dock Dummbo, an impromptu meeting is established where the whiteboard drawings and audio are captured to facilitate the meeting. Conference assistant is another application that assists conference attendees to decide which activities to attend, provide awareness of the activities of colleagues, enhance interactions between users and environment, assist users in taking on presentations, and aid in retrieval of conference information after it is concluded. This application uses a variety of contextual parameters including location, time, identity, and activity. Cyberminder application built using Context toolkit supports the creation, delivery and handling of reminders. Augmented wheelchair uses context to improve word prediction for mobile and speech impaired users.

2.2.3.   Context Information Service

Pascoe, Ryan, and Morse's [35] Context Information Service (CIS) is yet another object oriented framework which supports context aware applications. It is platform independent, globally scalable, and provides shared access to resources. Core features of CIS include contextual sensing, context adaptation, contextual resource discovery, and context augmentation. [38] CIS is a layered service architecture CIS is a layered service architecture consisting of service components that include world, world archive and sensor arrays as depicted in Figure 2.3. These components are extensible and reusable.

The CIS architecture consists of four CIS service components: World Archive, World, Sensor Array and a Catalog. CIS component world consists of artifacts, states, sensors, synthesizers, monitors and catalogs. Sensor arrays collect raw sensor data while synthesizers aggregate contextual data from other artifact states. Monitors control the way the contextual data are directed from sensors or synthesizers to appropriate artifacts. Artifacts are made up of name, type and associates set of states. Example of an artifact is the location (state) of a person artifact named 'Bill'. CIS client's can access the state of any artifact in the world. [37]

CIS has been successfully employed in Aura project which uses the contextual services provided by CIS and augments it with remote execution mechanism and a mechanism to capture user's intent to provide a distraction free pervasive environment. Among the primitive application developed using CIS is an application for observation study on Giraffe.

2.2.4.   *Context Service*

Context service [12] provides a middleware infrastructure for context collection and dissemination. Architecture of context service is depicted in Figure 2.4. The architectural components of context service include a *dispatcher, configurable set of drivers and collection if utility components*. Utility components include context cache, work pacer, an event engine and privacy engine.
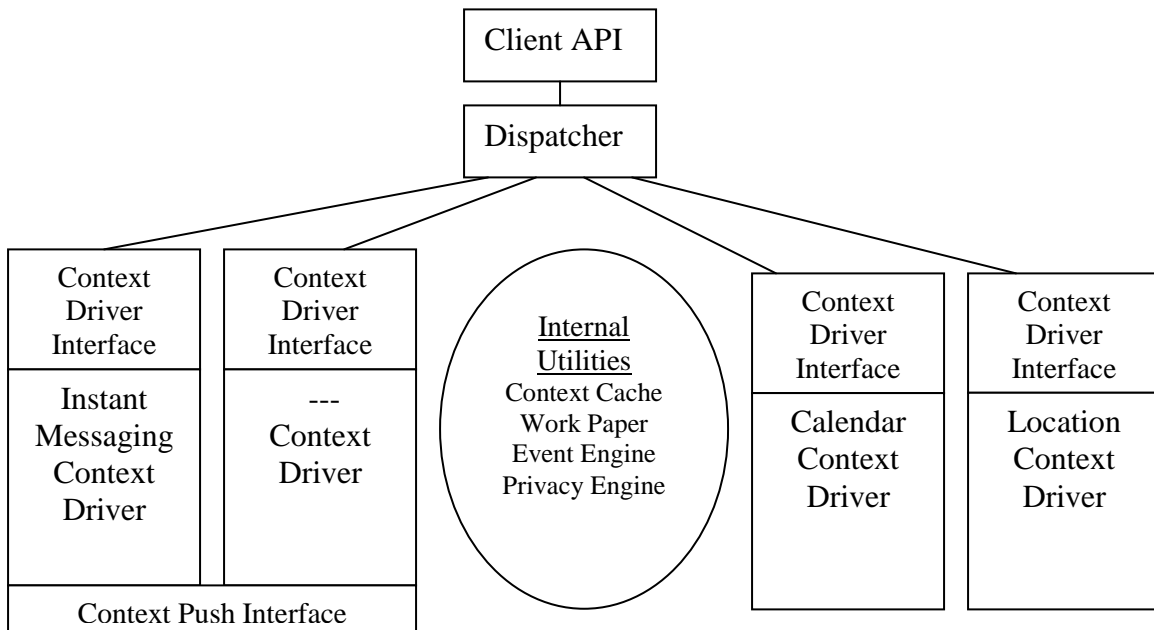
Figure 2.3: Architecture of Context Service [4]

Two applications built using context service illustrates its use in increasing the user experience: The notification dispatcher that uses context to route messages to a device

that is most appropriate to the recipient and a context aware content distribution system that uses context to envisage user's access to web content, and uses this information to preprocess and pre-distribute content to reduce access latency.

2.2.5.   *Activity Based Computing*

Activity based computing (ABC) framework [17, 21] developed at the university of Aarhus is a java based framework for development and deployment of activity based computing applications.
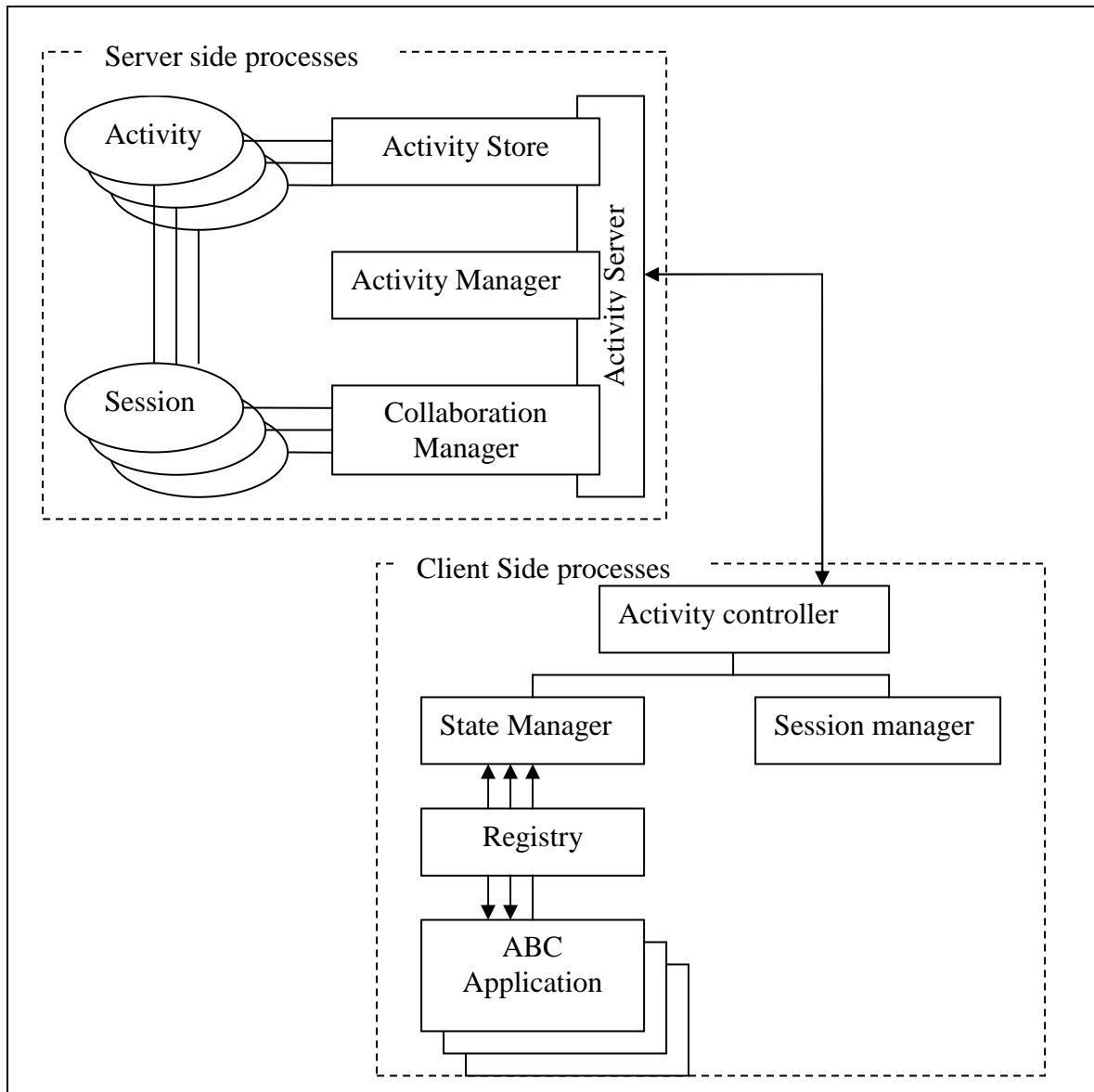
Figure 2.4: Architecture of Activity Based Computing Framework [32]

The infrastructure, as depicted in Figure 2.5, consists of a range of server processes and client processes where server processes run on one or more servers and client processes support the execution of the ABC applications. Activity state is saved centrally. Activity store provides an interface to manage (create, delete and retrieve) activities and keeps track of usage history of the user enabling the user to move forward and backward in the activity. Run time behavior of an activity is managed by activity manager by enabling activities to be created, initialized, paused, resumed and finalized by clients. Real time requirements for synchronous collaboration among active participants within an activity are handled by the collaboration manager. Collaboration manager manages a session object for each ongoing collaboration activity. Parties interested in changes to a session can add a session listener to the session. Client's activity controller acts as a link between the client and the server and it registers at one or more activity managers. It is notified about relevant events from the server processes. For example, if a user is invited to participate in an activity, it is notified and an appropriate signal can be made to the user. When the activity controller chooses to activate the activity, local state manager is notified which uses service registry to lookup appropriate ABC application. Session management is explained in Figure 2.6. Collaborative aspects and security provision by user authentication make activity based computing unique with respect to other similar frameworks.
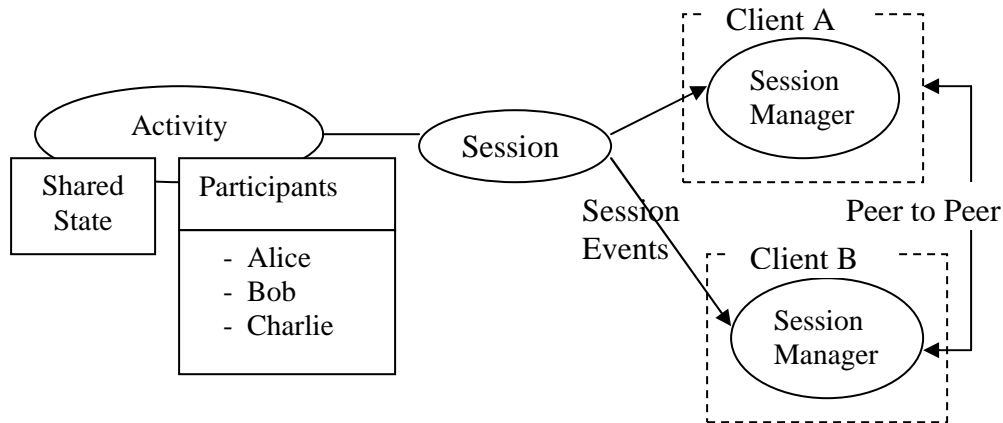
Figure 2.5: Session management in Activity Based Computing
Available form http://www.daimi.au.dk/~bardram/abc/abc.presentation.pdf

The idea behind collaborative aspect of ABC is that an activity may invite other people to join it and an activity becomes similar to a 'shared desktop' where all the persons in the activity view the same thing. Location monitors of the infrastructure were used to experiment 'proximity based login' where user is authenticated based on the proximity to the infrastructure.

2.2.6.   Task Based Computing

Aura [14] project uses Task based computing (TBC) framework and while it is similar to activity based computing, it provides a wider scope than ABC project. Aura leverages CIS to enhance the contextual features provided by CIS to make it more suitable in pervasive computing environment. [40] Aura deals with more levels than ABC

computing. The main objective of Project aura is to address the issue of the limited resource of human attention. Human attention is a scarce resource especially in a pervasive computing environment involving use of handheld devices, wireless communication, and smart spaces. The scarcity of human attention results form the fact that humans are often preoccupied, performing multiple activities at the same time. Intermittent network connection and limited resources of mobile resources add to the challenges of having *distraction-free pervasive computing*.
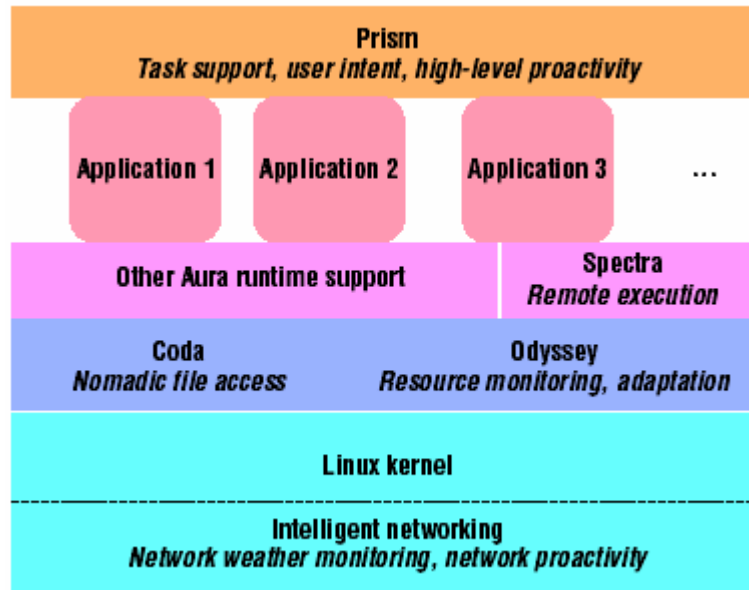
Figure 2.6: The Aura Architecture [14]

The two broad concepts aura applies are *proactivity and self tuning*. Proactivity is the system layer's ability to predict requests from a higher layer. Aura's self tuning ability allows the layers to adapt by monitoring the demands made on them and adjusting their performance and resource usage characteristics accordingly. Aura combines knowledge from different layer, interprets and uses it to make adaptations to the system's behavior. The aura architecture is shown in Figure 2.7. Coda and Odyssey components have been developed much earlier than aura but are being modified extensively to support pervasive computing requirements. Resource monitoring and application aware adaptation features are supported by Odyssey. Support for nomadic, disconnected, and bandwidth adaptive file access are provided by Coda. Spectra is an adaptive remote execution mechanism that uses context to decide how to best execute the remote call. User intent is captured and managed by Prism. It provides high level support for proactivity and self tuning.

2.2.7.   *Common requirements of context aware frameworks*

From the functionalities of the frameworks studied above, we came up with the common set of requirements that any context aware framework satisfies:
1. Sensor technology to capture the contextual information: Acquire raw contextual information

2. Support for event based programming model so as to have the ability to trigger events when certain context change is observed.
3. A way to communicate the sensed contextual data to other elements in the environment and a way to interpret the collected data: provide interpreted context to application.

2.2.8.   Summary and Comparison of frameworks

A summary of frameworks, applications developed using these frameworks and their architectural components are shown in Table 2 and Table 3. Comparison of context awrae frameworks is shown in Table 4.

Framework	Applications
JCAF	- context aware hospital bed - context aware pill container - context aware home automation system (CAHAS)
Context Toolkit	- In Out board - an information display that shows a user standing in front of it, a URL related to the research group that they are in - Dummbo, an augmented whiteboard - A conference assistant - CybreMinder - Applications for Aware Home - Augmented wheelchair
CIS	- Observational Study on Giraffe - A part of project Aura
Context Service	- Notification dispatcher - Pervasive content distribution
ABC	- A ward round - Distributed radiology conference
TBC	- Bandwidth advising scenario - Presentation preparation scenario

Table 2: Summary of applications implemented by context-aware frameworks

Context Information System (CIS) [6], the trivial Context System (TCoS), and the Secure Context Service (SCS) [7], support the common set of requirements of context awareness listed earlier. However these are all client server architectures and are not loosely coupled, distributed. Context Toolkit is closely related to JCAF in the kind of features it provides since Context Toolkit also provides a distributed and loosely coupled infrastructure. As of now, compared to other frameworks such as Context Toolkit, JCAF is less documented and has been employed in very few applications. Rome system[8] developed at Stanford is also closely related to the JCAF, however, Rome's *context trigger* is decentralized and the end devices are expected to have the capability to sense and handle all of its contextual information. With respect to securing context information, Secure Context Service is a step ahead in providing security and privacy in ubiquitous

environment. In SCS, identity of all the clients is known a priori and its security is based on role based access control. JCAF has less stringent security features which allow unknown client to provide context information. This accommodates the initial purpose of JCAF to have the framework for teaching and research purposes by providing building blocks for experimenting with context awareness.

Fram ework	Architectural Components
JCAF	Major components of JCAF RI Architecture: - Context Service - Access control : Ensure proper authentication for incoming requests - Remote Entity Listener - Context Client - Context Monitor
CT	- Sensors - Widgets - Interpreters - Context Server
CIS	- World Archive - World - Sensor Array - Catalog
CS	- Dispatcher - Configurable set of context drivers - Collection of utility components: utility components are a context cache, a work pacer, an event engine and a privacy engine 3 programming interfaces: - The client API, - Context push interface - Internal context driver interface
ABC	ABC runtime infrastructure is made up of: - Server side processes: Activity, session, activity server, Activity store, activity manager, collaboration manager - Client Side processes: Activity controller, state manager, session manager, service registry
TBC	Four Architectural components have been used - Task Manager called Prism - Context Observer (Uses CIS) - Environment manager - Suppliers

Table 3: Architectural components of context aware frameworks

Context system proposed by Shilit et. al [from the related work of context service paper] consists of a collection of environment servers, each providing information on one aspect of the context (e.g., a room, a meeting, a user, a work group). Server organizes context data as a set of name-value pairs and the applications can either receive the entire set or

can subscribe to any value changes in the set. Interfacing between the context system and context sources has not been considered. Privacy issue is also not addressed and context information is treated as available without any restrictions. Context toolkit provides lower level programming abstraction. Application developers are burdened to discover the context widgets, context interpreters, context aggregators, and discoverers.

	QoS DREAM	SALSA	CAHAS	PPS*	DRC[+]	Conference Assistant
Framework	DJINN	None	JCAF	TBC	ABC	Context Toolkit
Context Parameters	Location	Location, time, role, device	User, location objects	Location, network bandwidth	Activity	Time, identity, location, activity
Collaboration	No	Yes	No	No	Yes	Yes
Limitations	Framework itself demands for expensive equipment	Agents are not mobile	Requires training by the user			

Table 4: Comparison of context aware applications

*PPS: Presentation Preparation Scenario
[+]DRC: Distributed Radiology Conference

Support to build context aware applications is provided by various infrastructures indicated above among which the two promising supporting infrastructures for our thesis are: The Context Toolkit and the Java Context Aware Framework. JCAF would have been a better choice for our thesis, since it supports an Application Programming Interface (API), which can be used directly to handle contextual changes and since it is based on deployment in an organization. However, it is relatively new and is under active development. Being a relatively new proposal, JCAF has no proper documentation as of now. Context toolkit is relatively well established and the first version is now publicly available. Hence, modified context toolkit is a better choice for use in the framework proposed in this thesis.

2.3 USER INTERFACE GENERATION USING MODEL BASED UI DEVELOPMENT ENVIRONMENT

Model based user interface design and development has received much attention. One of the reasons for this attention is its usefulness in design of multi context user interfaces. Having multi context user interfaces is a direct requirement of most of the ubiquitous applications. Using model based approach allows the designer to focus on underlying models rather than the appearance of UI itself. Model based design allows for reuse of code.

Task modeling is useful for modeling tasks that belong to multiple contexts of use. Task model describes various tasks associated with an interactive system and the relationship between the tasks. There are lots of techniques in the literature that specify a task model. Some of the task modeling techniques include CTA, *hierarchical task analysis* (HTA), *Goals, Operators, Methods and Selection rules* (GOMS). Groupware Task Analysis (GTA), Task Knowledge Structure (TKS), Concurrent Task Tree (CTT), DIANE+, Method for USability Engineering (MUSE), Task Object Oriented Description (TOOD). Table 6 shows projects related to model based user interface development environments, organizations who created these projects, models supported, tools used to support this design, and final output generated using these tools.

CTT notation [46] is a widely used technique to model tasks. It defines four types of tasks: User Tasks which might be cognitive/ perceptive, Interaction Tasks which represent tasks that involve user interaction with the system, Application Tasks which are performed by the system, and Abstraction Tasks that refer to complex tasks. Icons representing each of these tasks are shown in Table 5.

Type of task	Icon
Abstraction	
Interaction	
User	
Application	

Table 5: Icons used to represent the four types of tasks defined in Concur Task Tree notation

Task model in CTT is represented as tree. Relationship between the tasks is either hierarchical-defined between parent and child or temporal-defined between adjacent siblings. CTT distinguishes and separates the context sensitive tasks from non-context sensitive tasks using various approaches: *monolithic approach, graph oriented approach, context sensitive separation approach, complete separation approach.* All the approaches represent the task model in form of one or more hierarchical structures – trees where each node of the tree represents a context sensitive task, a non-context sensitive task or a special node. Monolithic approach uses elliptical dotted line to surround the part of the tree that is context sensitive.

Project	Organization	Supporting models	Tools Used	Availability	o/p code
ConcurTaskTree (CTT), TERESA	CNUCE-CNR, Italy	Task modeling	CTTE	Tool is downloadable Not open Source	XHTML, VoiceXML
TIDE[1]	Virginia Tech, USA	Task model	TIDE	Under progress. Not publicly available	UIML to any supporting language
ARTStudio	CLIPS-IMAG Laboratory, France		ARTStudio	Not publicly available	Java, HTML
Mastermind[2]	Georgia Tech University of South California, USA	Task and presentation	Uses Dukas Task modeling Tool	Not available for download	C++ (Can run only on UNIX based systems)
MECANO interface model, MOBI-D	Stanford University, Redwhale Software	User, task, domain, presentation, design dialogue,	U-Tel, MOBI-D model editors, TIMM,MOBILE	Not available for download	XIML to HTML or WML
UMLi[3]	Stanford University, USA University of Manchester, UK	Considers class, use case, collaboration, deployment, and user interface diagrams	ARGOi ModelIng tool.	Tool: available for download not open source. Can be obtained upon request	Java Swing (Complex to learn)
DIANE	Université des Sciences et Technologies de Lille, France	Tasks	Tamot task modeling tool	Not applicable. Only provides formalism.	Only task modeling tool. No UI generation
FUSE[4]	Technische Universitat Munchen Siemens AG, Germany	domain, task, user model and dialog and layout guidelines	None	Not applicable (last work reported about this project 1997)	Only modeling tool
TADEUS	Universitat Rostok, Germany	Task, Object, User, Dialogue modeling	TADEUS Tool	Tool not available for download (98)	No information available
Teallach	University Glasgow, University Napier, university f Manchester, UK	Task model. Not UML	-	Tool concentrates on generation of ui for db app.	-
ADEPT	Queen Mary & Westfield College, UK	User, task, Abstract, concrete interface	-	No. (Last Work from this project reported in 97)	-
HUMANOID	University of South California. USA	Task, presentation	Template Editor	Not available. Pre project of Mastermind. Last work: 94	C++

Table 6: Model based User Interface Development Environments

---

[1] Source: http://perez.cs.vt.edu/publications/2002/CADUI-final.pdf

[2] Source: http://www.isi.edu/isd/Mastermind/Papers/ehci95.ps, http://www.isi.edu/isd/Mastermind/mastermind.html~

[3] Source: http://www.cs.man.ac.uk/img/umli/

[4] Source: http://wwweickel.informatik.tu-muenchen.de/forschung/ui/ui.html#FraLo

The distinction between context sensitive and non-context sensitive nodes is made more visible in graph oriented approach where sub trees resulting from context sensitive nodes are represented as separate trees and are related to the originating tree when needed. These approaches lead to cluttering due to introduction of additional relationships; context sensitive separation approach avoided this problem by introducing special nodes called decision nodes that connected the context sensitive sub trees. Complete separation approach further optimizes the representation of task model by factoring out similar sub trees that might possibly be located in different locations in the global task tree.

Information contained in the task model expressed in ConcurTaskTrees notation can be used to identify the presentations of an interactive application. Temporal relationships in the task tree are used to identify group of task that can be enabled at the same time. Enabled Task Set (ETS) are sets of tasks that can be enabled over same period of time. Temporal relationships and constraints indicated in task model play major role in computing ETSs from task tree. Tasks in single set form a part of single presentation. Heuristics (rules) can be applied to merge two or more enabled task sets into Presentation Task Set (PTS). This helps designers to reduce number of presentations in the final user interface.

2.4   ADAPTIVE USER INTERFACES

With increasing number of different types of devices and their use in ubiquitous environments, adaptation of user interfaces has become a necessity rather than a facility. Adaptation of the interface occurs when there is change in the "context". We classify contextual parameters into three categories: user, environment and platform. User category includes parameters such as user preferences, user role, and user behavior. Environment category constitutes of parameters such as task, ambient and aural conditions, other user's in the vicinity. Platform category includes the device's display characteristics, operating system, and available interaction mechanisms.

In this section, we review challenges related to adaptive user interface languages, related terminology, and quality metrics to measure the quality of user interfaces, tools and languages used to develop user interfaces. We further review various frameworks and toolkits that have served the purpose of adaptation of user interfaces, some of the projects that illustrate the adaptive features of user interfaces. Next, a review on various XML based User Interface Languages is provided. The section concludes with a comparison of these frameworks, toolkits and languages based on the features they provide.

2.4.1.   *Terminology*

Cross Platform or platform independent User Interfaces: These User Interfaces have their UI code portable and hence can run on multiple operating system platforms.

Multi device user interfaces: These user interfaces will allow a user to interact using various kinds of computers including desktops, laptop, palmtop, PDA with or without keyboards and mobile telephone.

Multiple user interfaces: These provide different views of the same information and co-ordinate the services available to users from different computing platforms. Computing platform refers to hardware, computing capabilities, operating system and UI toolkit.

Multi model interactive User interfaces: The User interfaces that provide support for multiple ways to interact are called multi model user interfaces. A user interface is said to implement multimodel user interactions when it has the ability to adapt the interactions based on input and output capabilities of the devices. For example, limited display size may demand for speech recognition and speech synthesis technologies.

Universal user interfaces: "These UI can support a broad range of hardware, software and network capabilities with the central premise of accommodating users with a variety of characteristics. These characteristics include diversity in skills, knowledge, age, gender, disabilities, disabling conditions (mobility, sunlight, noise), literacy levels, cultures, income levels, etc." [43]

Context of use of an interactive system contains the people who use the system, the platform used to interact with the system and the physical environment where the interaction takes place.

Multiple User Interfaces: User interfaces that have characteristics of multi device, multi user and multi model interactive user interfaces.

Model based User Interfaces: In model based user interface design methodology, user interface is described using various models. The related models are task, domain (business object model), user model, dialogue model, presentation model. [86] provides the following definitions "*A **task** model describes static and dynamic organization of work. A **user** model characterizes users and specifies their perception of tasks and organization of work, access rights to data, and their preferences for interaction modalities. A **business-object** model specifies objects of the problem domain with attributes, methods and relations, as well as the behavior of these models. A **dialogue** model describes the structure and behavior of interaction devices, features, and modalities. A **presentation** model extends the dialogue model by graphical representations of its elements.*"

Transcoding: Transcoding is a technique used to adapt web content for increasingly diverse kind of devices that are being used to access web pages. Transcoding requires transformation of the web page to convert into suitable format for a particular platform. A simple transcoding approach is to associate semantics with the structure of web page and perform the transformation based on these semantics. A more complex approach is to associate annotations on structural elements of the web page and use annotations to perform transformation.

2.4.2.   Quality Metrics

In this section, we present quality metrics of user interfaces, user interface languages and tools.

2.4.2.1.   Quality metrics for user interfaces

Plasticity: *Plasticity* is defined as "capacity of an interactive system to withstand variations of context of use while preserving usability" [42]. Plasticity can be described by the set of contexts it is able to accommodate. Metrics to evaluate plasticity of user interface include Plasticity threshold, size, shape, cardinality and topology of surfaces, context frequency and migration cost. Size, shape, cardinality and topology of surfaces are useful metrics for reasoning about the plasticity of a particular technical solution while context frequency and migration cost are metrics that relate to user's perspective.

Plasticity Threshold: Contexts at the boundaries of a set (of contexts) define Plasticity threshold of the UI for this set. It characterizes system's capacity of continuous adaptation to multiple contexts.

Overall quality metrics for plasticity: "The sum of the surfaces covered by each set, or the sum of the cardinality of each set, defines an overall objective quantitative metrics for plasticity. In other word, this sum can be used to compare solutions to plasticity: A user interface U1 is more plastic than a user interface U2 if the cardinality of the set of contexts covered by U1 is greater than that of U2." [41]
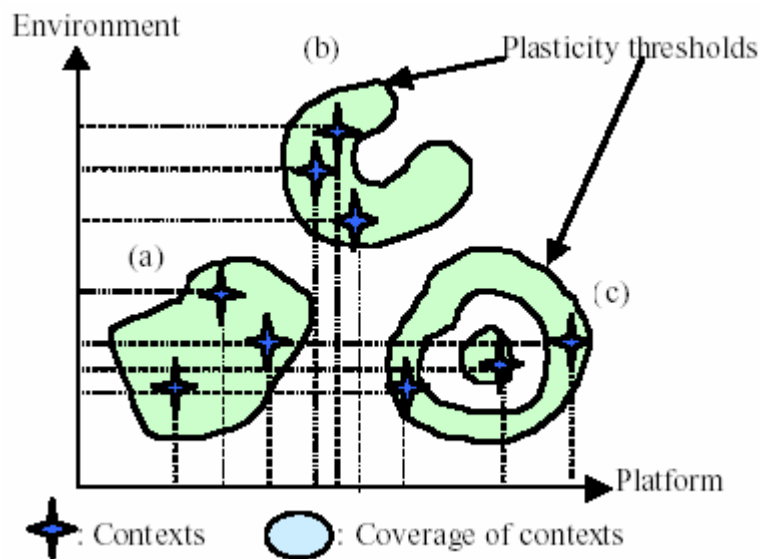
Figure 2.7: Measuring plasticity from system's perspective [41]

Size of largest surface: large surfaces denote a wide scale of adaptation without technical rupture.

Cardinality (Number of distinct sets): A large number of sets indicate presence of multiple sources for technical discontinuities. For example, GSM does not work everywhere which leads to discontinuities on platform axis. User interface designers have to check if these discontinuities are expected by the users.

Surface shapes: A convex shape represents a comfortable continuous space (figure 2.8(a)) and concave surfaces (figure 2.8(b)) raise design issues. Typical ring shape interface (figure 2.8(c)) indicates that the interior of the rings are not covered by the user interface. It indicates a technical discontinuity for contexts that are contiguous. The design decision should be made based on whether the user is expecting the tiny rupture in context coverage, and whether this inconsistency is a problem from user's perspective.

Context Frequency expresses how often users will perform their task in a given context. Designers have to revise technical solution space if large surfaces correspond to less frequent contexts and/or multitude of small surfaces is related to frequent contexts.

Migration cost: It characterizes the user's tolerance to context changes. Migration cost measure the physical, cognitive and conative efforts [46] users have to pay when migrating between contexts, whether these contexts belong to same or different surfaces. If users are expected to move between contexts that belong to different surfaces, discontinuity in usage system will be perceived. The designer may either choose to reconsider the deign process to avoid migration cost or stick to their solution for well-defined reason in which case observability of the technical boundaries should be the focus of special attention in order to alleviate transition costs.
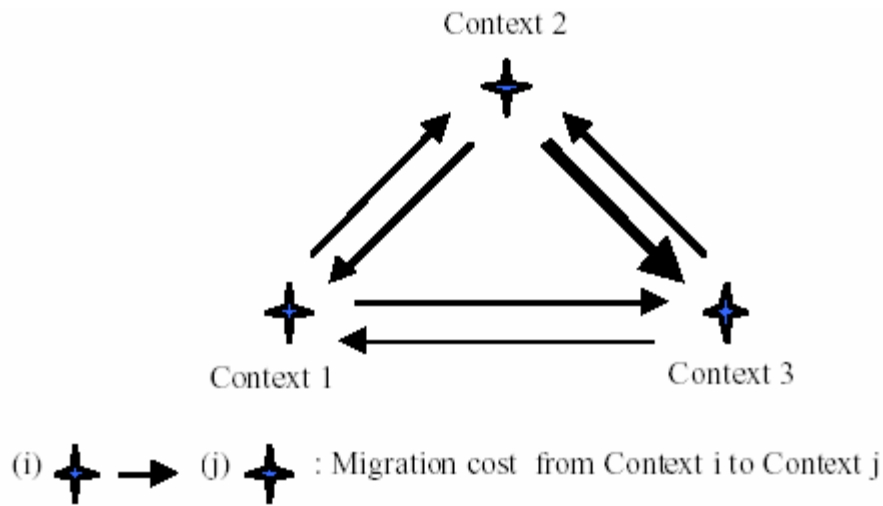
Figure 2.8: Measuring plasticity form user's perspective [41]
Arrow depicts the capacity of migrating between contexts and the thickness denotes human cost.

2.4.2.2. *Quality Metrics of user interface languages and tools*

Expressiveness, readability, evaluation, compilation/interpretation, knowledge representation, manipulation, threshold, ceiling, path of least resistance, predictability, and moving targets are the various metrics using which UI languages and tools can be measured [9,12].

Threshold of a user interface language relates to the learning curve involved in using the language. For example, a language that requires use of multiple languages or requires high learning curve is said to have greater threshold. Ceiling refers to the capabilities of user interface development tools. Tools useful for only class of user interface applications have low ceiling. Predictability, as the name suggests, is the measure of forecasting which involves the developer to expect or predict what to modify at a high level notion to produce a desired change.

2.4.3. *Adaptation to context – Architectures and Toolkits*

2.4.3.1. *Plastic user interfaces*

We explain here the five step process to perform plastic adaptation and then review the available tools that can help in achieving plastic user interfaces.

Plastic adaptation is structured as a five step process: *System detection, system identification, system selection, system transition, system execution.*

Adaptive plasticity, adaptable plasticity, mixed plasticity: The five steps of adaptation process can be performed by system only, users only or mixture of users and system. In the first case, system is said to be capable of adaptive plasticity, in the second case, system supports adaptable plasticity; and the third case system supports mixed plasticity.

System Detection involves detection of the conditions for adaptation. With reference to plasticity, it refers to variations of contexts of use. It requires the capacity of sensing physical environment and capacity of modeling physical platforms it is supposed to support.

System Identification deals with identification of candidate user interfaces appropriate to the new context of use. These may be computed on the fly, or selected from pre-computed set of user interfaces, or from a predefined set of ad hoc user interfaces.

System selection is selection of user interface and it relies on a problem solving strategy. User may assist the system in this task.

System Transition involves transition from current user interface to newly selected solution.

System Execution deals with execution of the new user interface until next conditions to adaptation are met.

ARTStudio (Adaptation through Reification and Translation Studio) is a software tool for development of pre-computed multi-platform user interfaces. Resources of the platforms on which UI will be deployed are not known at the deign-time. Multi targeting is limited to Java enabled single screen platforms and web pages using a Java HTTP server. A UI build using ARTStudio can adapt to the platform characteristics at run time. Task modeling is done using ConcurTaskTree. Abstract UI is modeled as a structured set of workspaces similar to the task model. Every task has corresponding workspace and vice-versa. The platform model is a UML description that obtains the screen properties such as depth and size, and the programming language supported by platform. The interactor model specifies 'representational capacity', 'interactional capacity' and the 'usage cost'. The concrete UI uses the abstract user interface, the platform, interactor models and the heuristics. It consists of a set of mapping functions between workspaces and display surfaces such as windows and canvases; between concepts and interactors; between the navigation scheme and navigation interactors. [43]

2.4.3.2.   *XWeb*

XWeb addresses the problem of interacting with services by means of variety of interactive platforms. The idea is to have number of clients running on various platforms access XWeb services. Service creators are abstracted from interactive techniques/devices. It provides support for cross platform UI, cross model interaction, support for adaptation to screen size and support for both speech and visual interfaces in the same model.

XWeb is based on WWW architecture with additional capabilities of interaction and collaboration. XWeb servers provide responses to the XTP network interaction protocol. Just like World Wide Web defines HTTP protocol as basis for communication between client and server, XWeb defines XTP protocol.

XWeb defines a method to define user interfaces using the XView language. XWeb deals with string based interactors, including string representation of numbers. It can generate html form-like views of the data. XView specifies a number of standard user interface widgets called interactors, which has to be implemented on the different client platforms for XWeb applications.

XWeb defines an interactive protocol for communication between they client and the server, defines user interface sin a form that is independent of particular mode of operation (platform independent interfaces), adapts interaction to available input devices ranging from minimal button set to interactive room, adapting to variation in screen size and aspect ratio, defining interfaces that can be used on speech and audio as well as visual display.

2.4.3.3.   Odyssey, Coda

"Coda and Odyssey are experimental systems built by the research group of Professor M. Satyanarayanan in the School of Computer Science at Carnegie Mellon University to explore adaptation issues.   Both systems embody adaptation at many levels. Coda implements application-transparent adaptation in the context of a distributed file system.   Since the POSIX interface is preserved, legacy applications can run unmodified on Coda. In contrast, Odyssey supports a type of adaptation called application-aware adaptation that is better suited to multimedia data such as speech and video. This overview paper provides more details on these concepts." [16]

2.4.3.4.   Luxor – An XUL Toolkit

Luxor is a free, open source XML User Interface Language (XUL) Toolkit in Java. XUL was created for Mozilla application and is used to define its User Interface. Hence, XUL is used with Mozilla technology and it is used to build cross platform applications. The applications built using XUL are easily customized with alternate text, graphics and layout so they can be readily branded or localized for various markets.

XUL differs from other API based toolkits such as Swing and WinForms because it clearly separates the user interface into four parts – content (structure and description of UI elements), appearance (look & feel, skin, themes), behavior and locale (localization information for internationalization)

Key features of XUL: XUL is a cross-platform, widget based markup language based on existing standards. It separates presentation from application logic and provides ease of customizing for different customers or groups of users.

2.4.3.5.   PIMA

The project aims at device independent applications for ubiquitous computing. It uses a generalization process that allows the application development to be done either by a design tool or by abstracting a concrete UI. Generalization is done by reverse engineering the code of the UI. First, interaction elements are identified. Then, the properties and semantics of these elements are inferred. A specialized device with a device profile then creates another application specialized for a particular device. [55]

2.4.3.6.   Multimodal Interaction and Rendering System (MIRS)

Wolfgang et. al [14] has introduced an XML-based framework called Multimodal Interaction and Rendering System (MIRS). MIRS, a framework for multimodal interaction, is dedicated for limited, mobile devices and is based on the specification, exchange, and rendering of user interfaces and dialogs, which are specified by the means of Dialog and Interface Specification Language (DISL). DISL can be considered as a subset of UIML which is enhanced by the means of state-oriented dialog specification. DISL is based on ODSN (Object Oriented Dialog Specification Notation), which has been introduced to define user interface control by means of interaction states with transition rules.

2.4.3.7.   *Adaptive applications for ubiquitous collaboration*

Allan et al [10] provides a framework for development of applications adaptive to the client's computing platform. The framework supports adaptation of both shared data and user interface to user preferences and display characteristics.

One of the distinguished features of Allan's framework is support for collaborative applications. It provides application adaptation at run time and provides cross platform feature of UI by using the same technique as UIML which has abstract representation consisting of interactors and which later will be mapped to device specific representation.

To support collaborative applications, the system architecture is based on DISCIPLE infrastructure middleware [17] that supports intelligent data distributing and transformation of data to heterogeneous platforms, but has no support for adaptation of user interfaces. DIstributed System for Collaborative Information Processing and LEarning (DISCIPLE) is based on replicated architecture for groupware where each user runs a copy of the collaboration client, and each client contains a local copy of the applications that are foci of collaboration. It allows users to customize their workspaces by downloading software components and hence it has chosen replicated architecture instead of centralized approach for groupware design. DISCIPLE is organized into two layers – communication layer and GUI layer. Communication layer is called the collaboration bus and deals with real time exchange of events concurrency control, crash recovery, dynamic joining and leaving of sessions. The collaboration bus consists of a set of communication channels to which peers can subscribe and publish information. Generic application specification expressed in interactor language (containing interactors) is transformed to device specific application specification which is expressed in terms of interactive graph containing widgets.

Adaptation support is provided through the use of transformation description. Transformation description has the potential to hold information related to user preferences, device characteristics and available resources. Inclusion of other contextual information has not been addressed. As of now, this framework only supports application adaptation based on static information like static device characteristics. It specifies that the current framework can be extended to provide dynamic adaptation but does not specify mechanisms on how it can be done.

2.4.4.   *Projects illustrating adaptive user interfaces*

2.3.4.1.   *AVANTI Project*

AVANTI (AdaptiVe and Adaptable INteractions to multi-media Telecommunication applIcations) addresses the interaction requirements of disabled users using web-based multimedia telecommunication applications and services. The project facilitates the development of user interface of interactive software application that adapts to individual user abilities, requirements and preferences. The project developed a technological framework called "Unified User Interface Development Platform" for the design and implementation of user interfaces that are accessible by people with disabilities.

Components of AVANTI system include a collection of multimedia databases, the AVANTI server and the AVANTI Web browser. Databases are accessed thorough a common protocol (HTTP) and provide mobility information for disabled people. AVANTI server maintains knowledge regarding the users, retains a content model of the information system, and adapts the information to be provided, according to user characteristics (hyper-structure adaptor). AVANTI web browser is capable of adapting itself to the abilities, requirements and preferences of individual users.

2.3.4.2.   SEESCOA Project

The motivation of SEESCOA (Software Engineering for Embedded Systems using Component-Oriented Approach) project stimulated from embedded systems that are constantly migrating to emerging technologies. Its goals include separation of UI design from low level programming, ability to migrate UIs from one device to another while automatically adapting to new device constraints. The project seeks to adapt Component Based Development (CBD) technology. The idea was conceptualized to avoid the problem of redesigning UIs whenever new technology came into market. The experiments have used XIML as the user interface definition language. [81]

2.3.4.3.   CAMELEON

The goal of CAMELEON Project (**C**ontext **A**ware **M**odeling for **E**nabling and **L**everaging **E**ffective interacti**ON**) is to build methods and environments supporting the design and development of context-sensitive user interfaces in model based approach. There are four basic layers of CAMELEON framework (Figure 2.11).
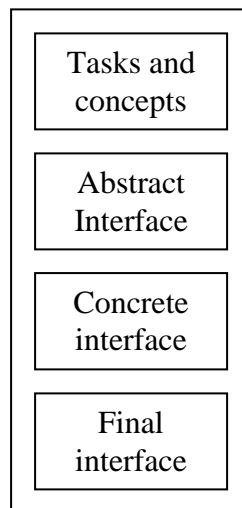
Figure 2.9: Four basic layers of CAMELEON Framework [91]

The tasks and concepts level abstracts the specification of UI by defining set of tasks and their relation to the system's functionality. Abstract Interface consists of set of presentation units where each unit supports execution of logically connected tasks. A

concrete UI specifies the user interface in terms of abstract intercators and their location. The final UI is generated form concrete interface where UI is described in code or programming language such as Java or HTML. ARTStudio is based upon this framework. In the CAMELEON Project several tools have been developed, two of them are publicly available: TERESA, VAQUITA.

a. VAQUITA

VAQUITA is a set of techniques established to reverse engineer UIs that were not designed according to a model-based approach to incorporate them in the same pipe-line and to allow migration of UIs from one computing platform to another. The current sub-goal of VAQUITA is to reverse engineer a web page onto a presentation model according to flexible heuristics. The flexibility of this process is of high importance considering the many design options that may have been decided at design time. Therefore, the process is guided by questions to the designer, information selection in the contents, and transformation alternatives.

b. TERESA

TERESA (Transformation Environment for inteRactivE Systems representations) produces different UIs for multiple computing platform from a generalized task model which is progressively refined for different platforms [55] general specifications expressed into XHTML code are mapped for each platform such as web, pocketPC, and mobile phones, using various presentation and dialog techniques. TERESA defines a UIDL called TERESAXML that supports several transformations including: task model to presentation task sets, task model into abstract UI, abstract UI to concrete UI, and generation of the final UI. TERESA is a tool for modeling task models in CTT notation. It has the capability to generate user interface in XHTML and voice XML from the CTT notation of task model. First, the task model is created using TERESA tool. This is used to generate enabled task sets ETSs that groups the tasks in such a way that all the tasks in the same group can be enabled at the same time. Heuristics may be applied to the ETSs to further group multiple ETSs into presentation task sets PTSs. User interface can be automatically generated from PTSs by the tool or user interface designer can intervene in the generation process to specify the presentation characteristics such as color, style, font, grouping, link types, etc.

2.3.4.4.   PALIO

Personalized Access to Local Information and services for tourists (PALIO) proposes a new framework that supports location awareness to allow the dynamic modification of information presented (according to position of user); adaptation of contents to automatically provide different presentations depending on user requirements, needs and preferences; scalability of information to different communication technologies and terminals; interoperability between different service providers in both envisaged wireless network and the World Wide Web; offer services through fixed terminals in public spaces and mobile personal terminals, by integrating different wireless and wired telecommunications technologies. Hence, PALIO framework is capable of adapting the content and presentation of services for use on a wide range of devices, with particular

emphasis on nomadic interaction from wireless network devices. Capabilities of PALIO framework in context of multiple user interfaces include device and platform independence, device and platform awareness, uniformity and cross platform consistence, general context awareness, and user awareness.

2.5   USER INTERFACE DESCRIPTION LANGUAGES

Variants of HTML (Hyper Text Markup Language) have been introduced to enable use of internet services on mobile limited devices: C-HTML (Compact HTML), WML (Wireless Markup Language), and MML (Mobile Markup Language). C-HTML is a subset of HTML which excludes all the elements that do not meet mobile device requirements like small display, low memory and slow CPU. Art of transcoding HTML to C-HTML involves not only replacing or deleting unsupported elements but also to restructure the document so that no information gets lost. WML is based on combination of HTML40 subset, HDML (Handheld Device Markup Language) and some extensions. In addition to HTML structures, WML is composed of set of cards, which correspond to states in user interaction dialog. MML is a family of languages defined by Japanese provider J-Phone: S-MML (Small MML), M-MML (Medium MML), F-MML (Full MML). These languages are defined for small devices with 4lines x 12 characters, screens with 160x 100 pixels and screens with larger resolution respectively. HTML and its variants provide very poor capabilities for the definition of graphical user interfaces. Languages defined are specifically tailored towards the device characteristics. Most of these languages are native and specific to the platform. These interface descriptions are useful part of adaptive user interfaces since several general UI description languages such as UIML can be translated to platform specific description languages such as HTML, WML, and Java. In this section we review various XML based user interface description languages that aid in creating adaptive user interfaces.

2.5.1.   UIML

User Interface Markup Language (UIML) is an XML language to define user interfaces in a device-independent manner. UIML is used to define interface elements such as buttons, menus, lists, and other controls. It also defines the layout and design of the controls, and actions to take place when certain events occur. Events may be created by the user interacting with the interface. Developing user interface using UIML involves writing UIML code which is a low level specification of the user interface. The advantages of using UIML include using UIML to describe the UI's behavior in a device-independent manner, its ability to give as much power to a UI implementer as a native toolkit, its ability to describe content, structure, behavior and style of UI separately.

cHTML provides a compact version of HTML and the features it supports considers least common denominator of multiple platforms. Unlike in languages such as cHTML, UIML supports all the features pertaining to a platform. This complete support is made possible by defining vocabularies. A vocabulary is a set of names, properties and associated behaviors for UI elements. Just like a programmer would use pre-defined libraries, UI designer can use pre-defined vocabularies to create user interfaces using UIML. UIML is a language to describe user interfaces for multiple devices; However, UIML does not provide any facility to write one description for multiple platforms. A UI designer has to create separate UIs for each platform using its own vocabulary.

UIML when combined with task modeling to provide a framework for multi-platform UI development. Task modeling enables UI designer to specify UI at a higher level of

abstraction than that supported by UIML. Task modeling is based on Concurrent Task Tree (CTT) notation [46]. Task model is mapped to generic UIML guided by the developer, where each task is mapped to one or more UI elements in the generic UIML. One generic UIML description can be written that can be mapped to multiple platforms that belong to a single family model. Two criteria that define a family model are the physical layout of the UI elements, and navigational capabilities provided by the platforms. A generic vocabulary is used in the family model. A generic vocabulary is a vocabulary defined for all platforms within a family model. A generic vocabulary of UI elements combined with UIML, can describe any UI for any platform within its family. Figure 2.12 describes the framework for building multiplatform UI using UIML and task modeling.
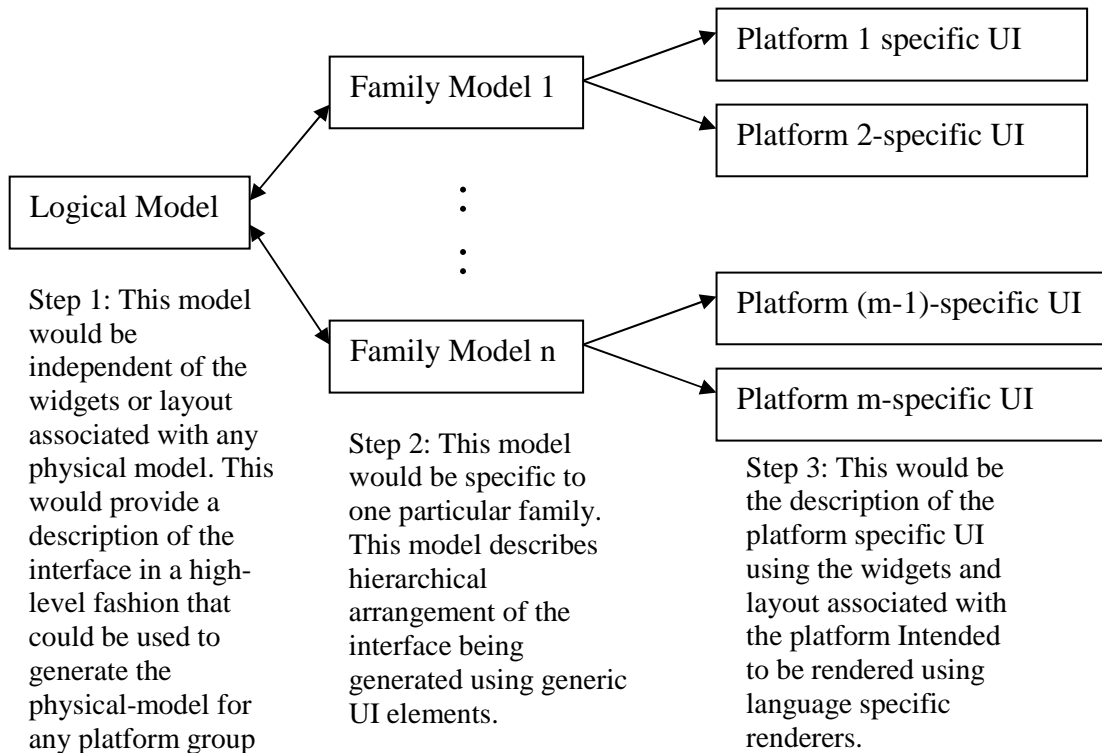
Figure 2.10: Framework for building multiplatform User Interfaces using UIML. [43]

Transformation-based Interaction Development Environment (TIDE) uses three levels of abstraction: UIML description with generic vocabulary, a platform specific UIML description, and the final UI. It allows the designer control the mappings between these abstractions. Task model is another abstraction that is being incorporated in TIDE. While UIML started with a clean paper to specify user interfaces and now extending its work for supporting task models. The objective behind using task models is to generate multi-platform user interfaces.

2.5.2.   XIML

eXtensible Interface Markup Language, maintained by XIML Forum – an independent Consortium, is an XML-based language for developing multiple user interfaces by transforming and refining user tasks and UI models. It provides representation framework for industry and universal support of functionality across interface lifecycle which includes phases of "design, development, operation, management, organization and evaluation" [45].

XIML is very abstract interface definition language which divides definition of interface into "components", high level building blocks of an interface. Examples of components include task(business process), domain(defines a hierarchy of components), user(defines hierarchy of end-users), presentation, and dialog(defines actions within the interface). XIML supports task modeling, that is, it has the ability to represent abstract concepts such as user tasks, domain objects, and user profiles. Components are mapped to "elements" which are concrete representations such as "widgets". XIML representation framework provides support for relational modeling between components.

XIML can specify any type of model, of model element, and relationships between. Although some predefined models and relationships exist, one can expand the existing set to fit a particular context of use. XIML has been used in MANNA for platform adaptation, in VAQUITA to support reverse engineering, and in Envir3D to transform graphical UI into a virtual one [55]

2.5.3.   AUIML

Abstract User Interface Markup Language (AUIML) [70], formerly known as Druid, is an intent based interface definition language, primarily developed at IBM. It allows UI designers to design the intent of the interface without tying the tasks to any concrete, device specific realization. AUIML consists of two major sets of elements – those that define data model and those that define presentation model.

Once the application is implemented, AUIML Toolkit can be deployed as a Java Swing application or as an HTML servlet without changing the application's code. The latest release of AUIML Toolkit includes both java Swing renderer and HTML renderer. Java Swing renderer provides rich client functionality to display the panels.

2.5.4.   USIXML

USer Interface eXtensible Markup Language (USIXML) is aimed at describing UIs with various levels of details and abstractions, depending on the context of use. USIXML supports a family of UIs including and not limited to device independent, platform independent, modality independent and context independent. Context sensitive UIs may be specified and produced from the USIML specifications. USIML supports multiple models for UI design such as task, domain, presentation, dialog, and context of use. Context of use in turn is decomposed into user, platform and environment. These models are structured according to 4 layers of CAMELEON framework: task and concepts, abstract UI, concrete UI and final UI. On May 15th, 2004, USIXML web site has been

hosted [90]. Work on USIXML was first published at Advance Visual Interfaces (AVI) workshop on May 25th, 2004 [43]

2.5.5.   AUIT

Adaptive User Interface Technology (AUIT) is the result of efforts to develop a new approach for the development of adaptable, web-based information system user interfaces. It provides developers with a set of device independent mark-up tags used to specify thin-client screen elements, element groupings, and user and user task notations. It shows how UI markup language can be interfaced with existing programming languages. AUIT augments current server side specification technologies like Java Server Pages and Active Server Pages instead of replacing them. Hence, AUIT scripts can be embedded in conventional Java Server pages to provide single adaptable thin-client interface for web-based systems.

At run time, AUIT uses a single interface description to automatically provide an interface for multiple web devices such as desktop HTML and mobile WML-based systems, while highlighting, hiding or disabling the interface elements depending on the current context." They have developed automated approach for splitting too large screens into parts for different display devices.

AUIT allows the developer to specify the screens independent of user, task and display device. To support AUIT implementation, a set of device-independent screen element tags have been developed for use within JSPs. AUIT screen descriptions are typically lower-level than those of conceptual web specification languages such as WebML and HDM.

AUIT has the added advantage of enhancing the capabilities provided by current server side implementations. It provides adaptation to user, user's task and device. It does not provide adaptation to other contextual elements such as environmental conditions and dynamic changes in user tasks. Also, AUIT has high ceiling, since use of AUIT for user interface development implies knowledge of server side development technologies such as JSP or ASP also.

2.5.6.   AAIML

Alternate Abstract Interface Markup Language (AAIML) [69] is XML based language for communicating abstract user interface definitions for a service or device to a user's personal device allowing it to act as a universal remote control (URC). Focus of AAIML is to enhance accessibility across devices and services in a manner appropriate to individual URC. This console will allow users to then interact with the target service or device using a tool that they are familiar to use. This is especially useful for making devices accessible to people with disabilities. URC can emulate the interface using whatever interaction technique that is applicable for that URC (point and click for desktop, hear and speak for phone, etc)

The language defines a set of "abstract interactors" for interaction. These have semantics and function but no concrete realization. These are mapped to concrete interactors much like UIML. The language is built on event modal, built on eventing capabilities found in Upnp and Jini/Java.

2.5.7.   XUL

XUL is maintained by independent consortium, which was first built by the Mozilla project for use in Mozilla web-browser. Its goal is to support different viewing capabilities for multiple computing platforms. It has received international attention due to its capability to support multiple platforms.

However, there are drawbacks of XUL. XUL is very limited in scope compared to UIML and XIML. The scope is limited to use with Mozilla browser/Netscape 6 browser and is used only for defining user interface elements of web page. A developer using XUL must combine XUL with JavaScript, XBL, CSS, and DTD. It requires the developer to know and understand all these technologies. XUL does not separate the metaphors from XUL specification. For example, XUL has elements such as <hbox> (horizontally oriented box), <button>, and <menupopup> which are knitted to traditional GUI metaphor. Mapping such widgets to an alternate, non-traditional device such as voice-browser is not only infeasible but also requires a rewrite of UI for alternate device.

2.5.8.   GITK / GIML

GITK, Generalized Interface Toolkit, is an open source adaptation infrastructure. GITK uses Generalized Interface Markup Language (GIML) that describes interface in terms of patterns such as "action", "data-entry/value-choice/single/limited" instead of concrete components such as "button", "scrollbar". This feature differentiates GIML from many other XML based UI description languages. The advantage of having such a representation is that the interface can be rendered on any device without making assumptions of its available interface components. This forms basis for multimodal interaction. [104]

2.5.9.   XAML

XAML is the primary method of creating Microsoft's 'longhorn' programming model. XAML is a declarative language that enables developers to specify a hierarchy of objects with set of properties and logic. It is employed for visual interfaces to define a layout of text, images controls, and shapes. An event can occur when users interact and the logic is also embedded in the XAML file.

2.5.10.   Others

XForms is a W3C initiative used to create cross platform user interfaces. It is a GUI Toolkit for X Windows system. It is based on Xlib and features objects such as buttons, scrollbars, menus, etc. XForms separate the purpose from the presentation of the form. Hence, forms created using XForms are richer and flexible than HTML forms. They

provide universal accessibility and allow the forms to be routed to multiple users and locations.

XICL: eXtensible User Interface Component Language is a tool for browser-based user interface development. It allows for component based UI software development. UI can be developed by using HTML and XICL components, HTML elements can be extended and/or grouped to generate new and more abstract components that can be reused. XICL document is translated using XICL translator before rendering the UI. The output generated by the XICL translator is a DHTML document (HTML and JavaScript).

useML, VRIXML, IM2L, CXML are among other XML based User Interface description languages.

CXML, USIXML have been developed by Quentin Limbourg, Jean Vanderdonckt, et al. for providing user interface description language for context sensitive user interfaces. They both use CAMELEON framework and CTT task notation to support multiple levels of abstraction. This UI description language was targeted for the UI designer. It captures the essence of user interface independent of physical characteristics of the user interface.

2.5.11.   *Comparison of UIDLs*

Due to plethora of user interface description languages it is sometimes hard to pick one. As shown in the comparison, not all are standards and among the standards also, goals of each description language differ. For example, XICL, a W3C standard, is intended for UI component development browser based software. Goal of XForms, another W3C initiative, is to express form-based UIs at a more abstract level than HTML. XIML is one of the promising description languages that is known for its interoperability qualities. However, it is protected by copyright by XIML consortium and is restricted by XIML license. Most of the other UIDLs are free. UIML, developed from plain sheet of paper, is cross platform UI. It has started supporting task model with its TIDE tool. UIML has no support for context model yet. USIXML, has evolved with support to represent different models. Table 7 shows the comparison of selected user interface description languages. It shows the models supported by each UIDL, whether it is a standard or not, the tool support it provides and its availability.

S#	XML Based UIDL	Schemas Available?	Models supported	Download able?	Standard	Supporting tool	Other complementary tools
1	Liquid UI / UIML[5]	Yes	Dialog, Presentation, Domain - (parts)	Yes	OASIS	Code generator	TIDE design tool. - not yet publicly available
2	XForms	Yes		Yes	Yes- W3C	Translator	None
3	XIML	Yes	Task, domain, User, Dialog, Presentation, Platform, Design	Yes	Yes	Rendering engine, Code generator Editor	CTT design tool - publicly available
4	USIXML	Yes	Task, domain, presentation, dialog, context of use	Yes	Independent consortium	Translator	GrafiXML -publicly available
5	XUL	Yes	Dialog, Presentation	Yes	No	Rendering engine	None
6	AUIML	Yes	Dialog, Presentation	Yes	No	Rendering engine	
7	XICL	Yes		Yes	Yes - W3C	XICL Translator	
8	XAML	Yes		Yes	No		
9	TERESAX ML	Yes	Task, presentation Schema grammar available	Yes	No	Code generator	TERESA design tool
10	SUNML	Yes		No	No		SUNML Editor Noah Editor WML Merging tool
11	AUIT	No	Task, User	No	No		

Table 7: Comparison of XML based User Interface description Languages

---

[5] Source: GITK http://gitk.sourceforge.net/links.html

CHAPTER 3. FRAMEWORK FOR CONTEXT AWARE ADAPTIVE USER INTERFACE GENERATION

In this chapter, we present the design decisions for our framework for context aware adaptive user interface generation, framework itself, scope within the framework and further proceed to describe design of two components of the framework: context server and modeling and interface server. As described in chapter 1, requirements for systems with adaptive user interfaces are many-fold. System should provide the ability to handle both input and output using multiple mechanisms. There should be support for multiple metaphors and multiple platforms. Support for automatic adaptation of interfaces to various contexts needs to be provided.

3.1   DESIGN DECISIONS AND APPROACH

The goal is to have a framework that can generate user interface dynamically from the context specification and design specified using modeling techniques with focus on specifying and transforming context to user interface description. The design decisions for the framework are based on two aspects. First is context categorization. Second aspect answers what, when, where and how of context.

3.1.1.   *Context categorization*

Our definition of context here follows that given by Dey and Abowd [8, 9]. The set of contextual parameters is not limited. Examples of contextual parameters include device used, user role, user activity, user preferences. Each parameter poses different set of requirements for the system to support these contextual parameters. Categorization of these contextual parameters will make addressing the various contextual parameters with varied requirements.

Our approach is to categorize the problem space for these varying requirements of various contextual parameters into four classes and address the requirements of each class of problem space to build a unified framework for context aware adaptive user interfaces.

We classify contextual parameters into two categories based on the nature of changes: Static and Dynamic. Examples of static contextual parameters include platform, role, and user preferences. Platform includes the hardware, software, display and interaction capabilities of the device used to access the application. Figure 3.1 depicts this classification which is based on nature of adaptation of user interface.

Static contextual parameters can be handled during the design phase of user interface development while solution for handling dynamic contextual requirements requires runtime adaptation support. Examples of dynamic contextual parameters include location, network bandwidth, and time. Dynamic contextual parameters need to be handled at the run-time of user interface generation. If the contextual parameter changes over time during a single session, it is categorized as a dynamic parameter; else it is categorized as a static parameter.

Contextual Parametets

Static
Examples:
Platform
Role
User Preferences

Dynamic
Examples:
Location
Network Bandwidth
Time

Requires design time specification   Requires run time adaptation support
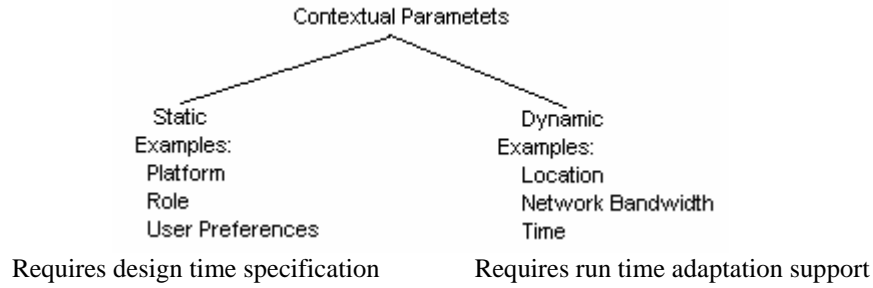
Figure 3.1: Classification of contextual parameters based on nature of adaptation

We identify two levels of adaptation to the application when context changes (Figure 3.2). The first level of adaptation occurs at the back end where functionality provided to the user changes with change in the context. The next level of adaptation occurs at the front end when presentation or user interface is affected due to change in context. Hence, another useful characteristic associated with contextual parameters is the layer of user interface adaptation they effect. Often, contextual parameters may affect both the layers; however, this categorization is significant since the solution that addresses adaptive user interface depend on it. Former level requires processing to be done at the information server which does the data filtering while the next level requires processing at the interface server which is responsible for user interface description and generation.

Contextual Parametets

Functional
Examples:
Role
Network Bandwidth
Time

Presentation
Examples
Platform
User Preferences
Location

Requires processing in information server   Requires processing in interface server
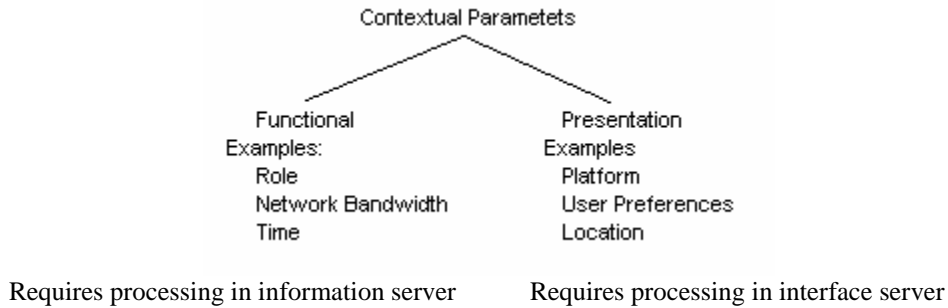
Figure 3.2: Characteristics of contextual parameters based on level of adaptation

Hence, we classify the problem space for having adaptive user interface into four classes based on level and nature of adaptation. Table 8 shows the contextual parameters that are generally associated with each class. The solution is based on these four classes of solution spaces. While a context parameter can be either static or dynamic, it can be mapped to multiple levels. For example, depending on the application requirements, platform of the user may be specified as a static context parameter. Also, the application requirements may indicate that it change in platform may effect content, layout, presentation, navigation and other facets of user interface. Table 8 shows two broad levels: content and non-content. However, the non-content level can in turn be made more granular by separating the navigational and presentation facets of user interface.

Level/Nature	Static	Dynamic
Functional And content	Role / identity (user's knowledge level, skill, background, goal),Platform, User preferences	Network Bandwidth, Time Location.
Non-content (includes Presentation Style, navigation, Layout/structure, Modality/interaction)	Platform, User Preferences,	Ambient conditions such as sound, lighting User activity (driving) User behavior Orientation

Table 8: Contextual parameters associated with the four classes of solution space for adaptive user interfaces

As described in Chapter 2, there have been number of attempts in describing context. We present a new contextual specification that is tailored for the use of adaptive user interfaces applications that adapt user interface to contextual changes. We incorporate, in the context representation, the characteristics of adaptive user interface: level and nature of adaptation which were defined in earlier section.

3.1.2.   *Where, When, What and How of Context*

How is context data gathered and used?
Context data can be gathered in three ways. Depending on the origin of context data, we define three types of context: Client context, sensory context and system aware context. Client context is the context that is communicated by the user to the server is called delivery context and it comprises of device capabilities/platform, user preferences, role and user behavior. Sensory context is gathered from sensors. Sensory contexts include location, ambient conditions such as sound and lighting, user activity. System aware context is the context that the server is aware of. It includes computational contexts such as network bandwidth, and data such as time.

There is a need for representation of client context and sensory context, so that these contextual data can be communicated to the *context server*. Composite Capability/ Preference Profile (CC/PP) is a W3C standard that represents user preferences and device capabilities for the purpose of supporting user interface adaptation to various user preferences and platforms. We propose an extension to this standardized profile, called Augmented-CC/PP (ACC/PP), to support other context data including user role and user behavior. To represent sensory context, we define a profile called *sensory profile* and it is sent to the context handler whenever new sensory data is available or whenever the update is considered as important.

Augmented CC/PP, sensory profile and system aware context are combined and analyzed to form the next stage of context representation called *user's current context data*. This

stage consists of static and dynamic contexts, both defined by the same schema; the only difference being the static or dynamic nature.

Gathered context data consisting of sensory context, delivery context and system-aware context is transformed to "***User's current context data***" XML documents that can be used by the adaptation infrastructure.

What information is associated with context?
Context data is associated with an identifier specifying unique user or object to which the current context data belongs and the level and nature of adaptation.

When is a change in context handled?
If nature of context element is static, then adaptation occurs during session initialization. If nature is dynamic, adaptation occurs at the run time and certain contextual targets trigger adaptation during the run time.

Where is the context change processed?
If level associated with the context is 'content', processing is done at the information server level. Else, if level associated with the context is 'presentation', processing is done at the interface server. A more granular level would have context, task, dialogue and presentation level which specifies the modeling component that handles the context change.

How does the application react to change in context?
There are two ways in which application can react to change in context. In some cases, we need active context awareness and in other cases, passive context awareness. [35] Active context awareness changes the environment without user's consent while the latter waits for user's consent so as not to disrupt user's current activity.

3.2   FRAMEWORK

Goal of the framework is to assist user interface designers and developers of interactive and ubiquitous software applications and provide a basis for adaptive user interface toolkit for creating context sensitive user interfaces. The framework, shown in Figure 3.3, consists of five major components: information server and business logic components, context server, context modeling tool, interface and modeling server, client device.

Context modeling tool: At the design time, a context model is created by the user interface designer using the context modeling tool. The tool provides a graphical user interface that allows the designer to specify various contextual conditions and the effect of context change on user interface. Our design of context model is based on Concurrent Task Tree (CTT) notation developed by Fabio Paterno [66]. Details of design of context model are discussed in next section.
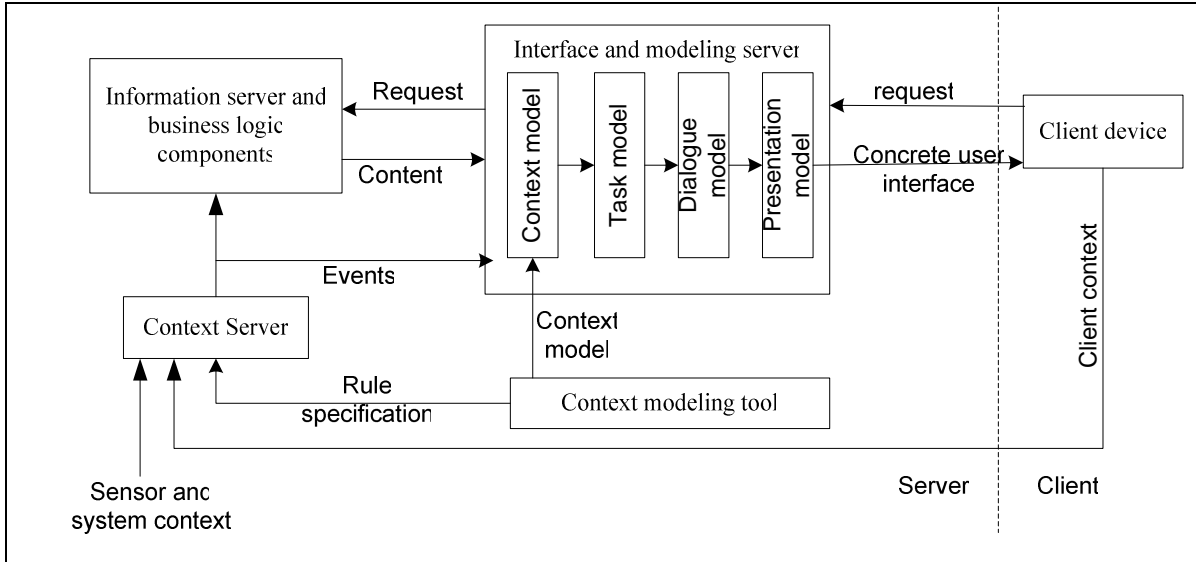
Figure 3.3: Framework for context aware adaptive user interface generation

Context server serves the interface and information server by hiding the details of collecting sensor information, abstracting, aggregating and interpreting data obtained from sensors and other sources into contextual information. Context specification is one of the inputs to context server. It specifies the context server which contexts the application is monitoring for. Context information originates from different places and it is of different types. Context server gathers client context, sensory context and system context and processes it. Events are triggered when the gathered context matches the context in context specification. Design of context server is presented in next section.

Interface and modeling server is responsible for adapting and generating the user interface. We consider four models: context model, task model, dialogue model and presentation model. Each provides a level of abstraction and the output of these models are context tree, task tree, abstract user interface and concrete user interface respectively. Context model obtained from the context modeling tool and events generated by context server are used to generate user interface dynamically at run time. Events specify the level at which regeneration has to start. If regeneration is at the context model level, task model is re-computed from the context model using current context. Other levels of user interface generation: generating abstract and concrete user interface follow. In some cases, events may specify that the regeneration has to start at the abstract level in which case, regeneration is started from that level. Design of context model and algorithms to process and transform context model to task model are presented in detail in Section 3.3.

Information server and business logic components handles adaptation issues related to content of user interface. It comprises of the business logic that handles the user's request. Response is sent to the user interface and modeling server which computes the user interface. If the application requires any change in the data and response itself, this component is queried again.

Client device is the originator of request and also sends information of client context to the context server. Format of client context is discussed in the design of context server presented in Section 3.3.

Scope of the thesis with respect to this framework is focused on *context modeling tool*, a part of *context server* and the context and task model components of *interface and modeling server*. Highlighted components of Figure 3.4 depict the scope of the current thesis.
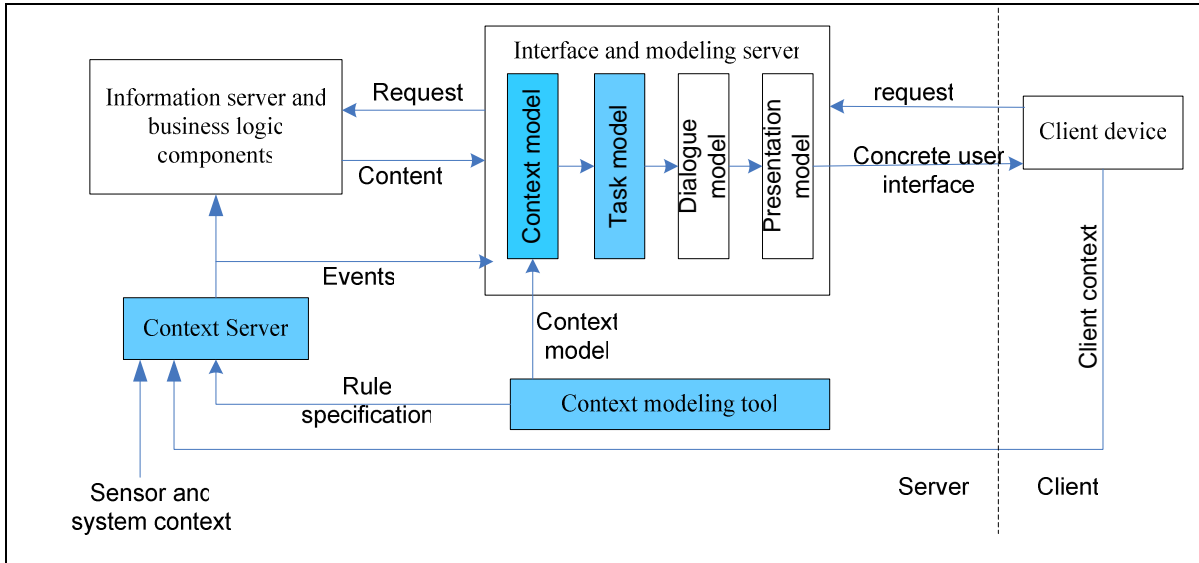
Figure 3.4: Scope within the framework

Figure 3.5 shows the information centric view of the framework. It depicts the information flow illustrating UI generation life cycle. ucc.xml and rules.xml specify the user's current context and rules respectively. They are the result of design performed by UI designer using context modeling tool. Result of context modeling is an XML file named cct.xml and is called concur context tree. Result of task modeling is a file named ctt.xml called concur task tree. Further down the user interface generation are abstract user interface and concrete user interface, which are the result of dialogue modeling and presentation modeling respectively.
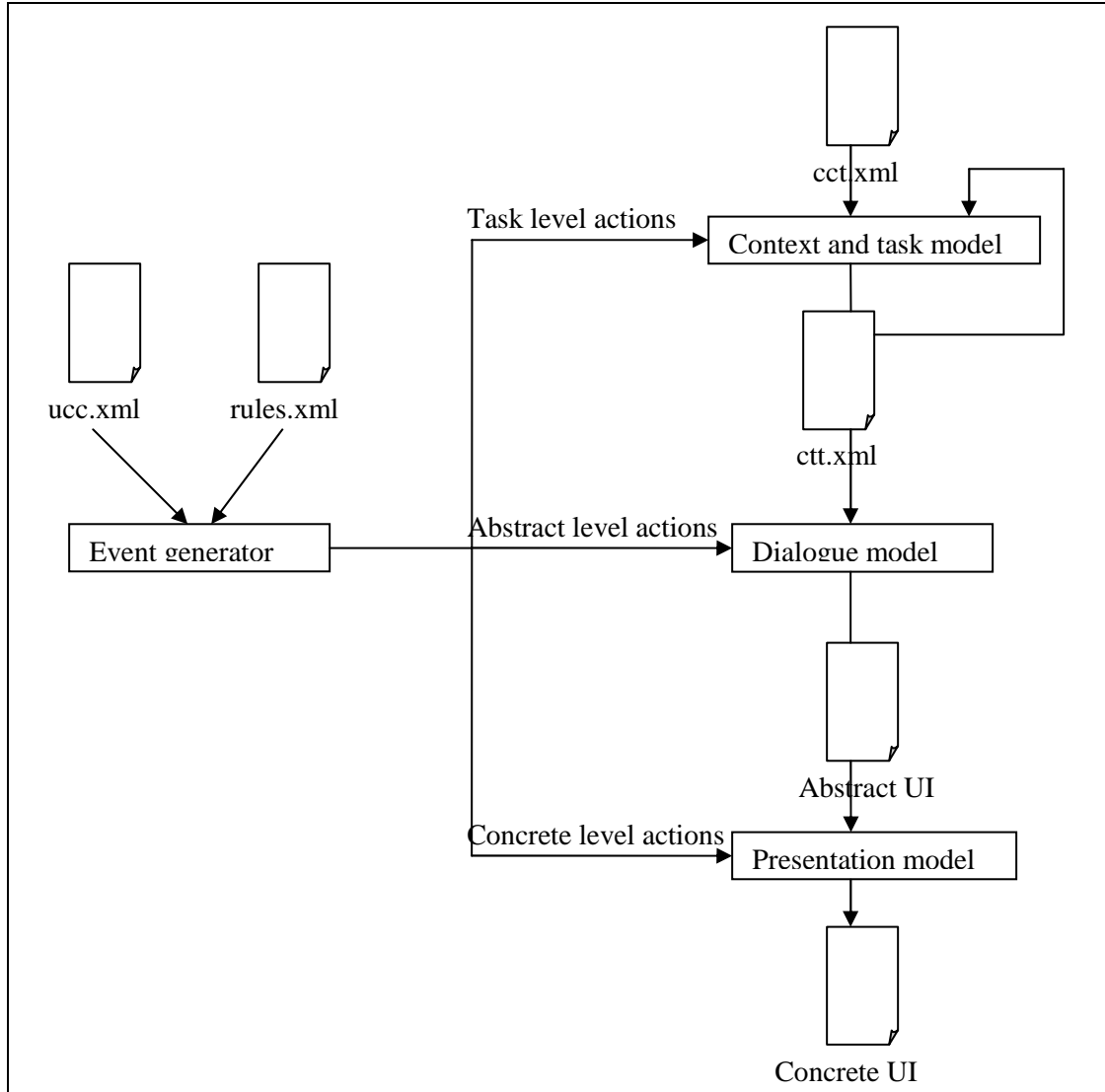
Figure 3.5: Information centric view of the framework

3.3   DESIGN OF THE CONTEXT SERVER

3.3.1.   Inside of context server

As the framework depicts, there are two kinds of input to the context server. One type of input is the raw context data. Second type of input is the rule specification. These two types of inputs correspond to the two major components of components of context server that include event generator and raw context processor. Raw context data arrives at the context server from three different sources: client context, sensory and system aware context, and context and action specification, sources being client, sensors and context modeling tool. Figure 3.6 depicts the major components in the context server and the data

flow between the components. The main purpose of context server is to abstract the way context is sensed and handled.

Raw context processor has similar responsibilities as context toolkit reviewed in Chapter 2. It abstracts the sensor information and context data, shares the context data and produces the data for each user. Output of raw context processor represents the user's current context named ucc.xml. User's Current Context (UCC) gives the detailed context specification, design of which is presented in the next section. rules.xml is the result of context specification and modeling done by the user interface designer using the context modeling tool. Event generator uses these two files to send events to other components of the framework. Parameter 'level' specified in rules.xml is used by event generator to decide the destination of the event. Destination is one among information server, context model, task model, dialogue model and presentation model.

Legend:
* Continuous data flow
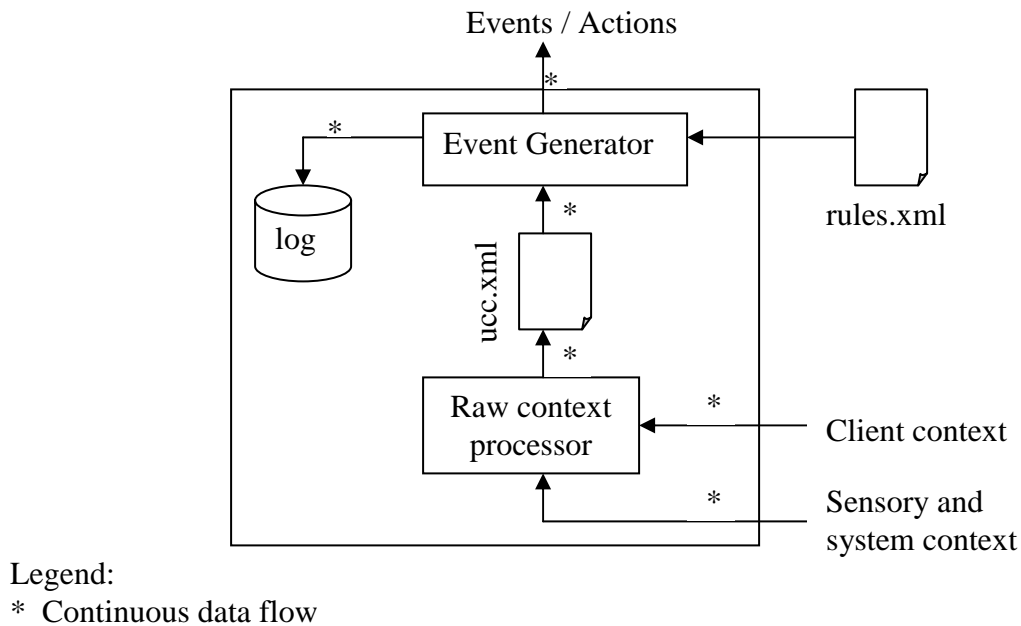
Figure 3.6: High level components and data flow in context server.

3.3.2. *Context Specification*

When user initiates session, user context data is sent to context server in form of augmented CC/PP. Context abstraction component of context server processes it with other contexts to send context data to context processor. Static context data is sent only during initialization. Dynamic data is sent periodically. Nature of each context parameter is communicated from context abstraction component to context processing component in the context server. However, level information for each context is computed at the context processing component using information obtained from context modeling tool. UCC document is used by the context processor with action specification to generate events to appropriate levels in user interface generation life cycle. We associate each context with user identifier to uniquely identify the user to which context is associated.

We also associate time stamp with each UCC document to inform context processor the time of creation of context. Context processor uses this information to maintain logs.

To specify context, our categorization of context includes plasticity type to make the specification logical. Hence, contexts are grouped into user, platform and environment. Context may or may not be associated with a parameter name. Each context parameter is associated with level, nature and value information. Nature and level of adaptation relate the context parameter to user interface adaptation requirements. Figure 3.7 shows the UML design of 'user's current context' schema. Each block in the figure shows the data type of the tag and the arrows indicate the tag names and their cardinality. Appendix A shows the complete schema of UCC.
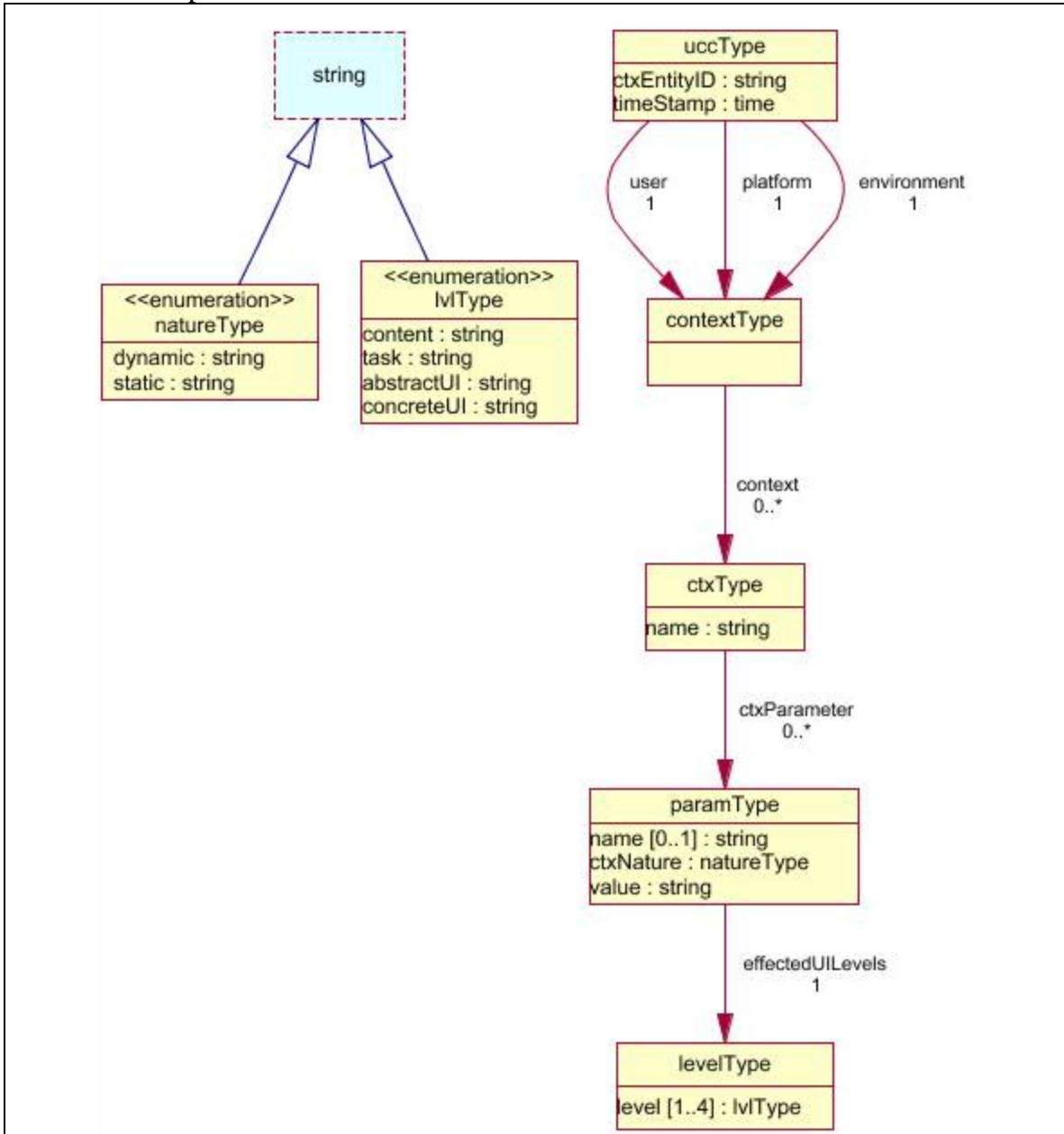
Figure 3.7: 'User's Current Context' (UCC) Schema Design in UML

Figure 3.8 shows an instance of the document showing representation of context that confirms to UCC schema definition in Figure 3.7. It shows the contexts being placed in

```xml
<?xml version="1.0" ?>
<UsersCurrentContext xmlns="http://www.w3schools.com"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="ucc.xsd" ctxEntityID="USRPat2140"
    timeStamp="12:20:46.275+01:00">
  - <user>
    - <context name="preferences">
      - <ctxParameter name="Navigator choice" ctxNature="static">
          <value>Graphic Link</value>
        - <effectedUILevels>
            <level>presentation</level>
          </effectedUILevels>
        </ctxParameter>
      </context>
    - <context name="taskFrequencyIncrease">
      + <ctxParameter name="Transportation" ctxNature="dynamic">
      + <ctxParameter name="Home Repairs" ctxNature="dynamic">
      + <ctxParameter name="Access Health Information" ctxNature="dynamic">
      - <ctxParameter name="Order Meals" ctxNature="dynamic">
          <value>4</value>
        - <effectedUILevels>
            <level>task</level>
          </effectedUILevels>
        </ctxParameter>
      + <ctxParameter name="Insurance Counceling" ctxNature="dynamic">
      </context>
    - <context name="activity">
      + <ctxParameter name="speedOfNavigation" ctxNature="dynamic">
      + <ctxParameter name="timeAfterLogin" ctxNature="dynamic">
      </context>
    </user>
  - <platform>
    - <context name="Device">
      + <ctxParameter ctxName="DeviceType" ctxNature="dynamic">
      </context>
    </platform>
  - <environment>
    - <context name="location">
      + <ctxParameter ctxName="latitude" ctxNature="dynamic">
      + <ctxParameter ctxName="longitude" ctxNature="dynamic">
      </context>
    </environment>
</UsersCurrentContext>
```

Figure 3.8: Sample context description

three logical groupings: user, platform and environment. The example also elaborates certain context parameters. This document keeps changing at real time. All the values associated with ctxParameter whose ctxNature is dynamic are the ones that change dynamically. As shown in Figure 3.6, an event generator monitors this document for match of predefined context values. Event generator uses rules.xml to find the matching context and also to determine the type and content of the event being generated.

3.4   **DESIGN OF INTERFACE AND MODELING SERVER**

The purpose of interface and modeling server, as the name suggests, is to walk through different models including context model, task model, dialogue and presentation model. Each model generates context model specification, task specification, abstract user interface and concrete user interface respectively.

Concur Task Tree model introduced by Fabio Paterno provides a formalism to represent user interactions with applications. It was not designed for ubiquitous applications which require that contextual parameters are the starting point of the design. [64, 47, 56] have discussed mechanisms to represent a context model on top of the task model to address this requirement of ubiquitous applications. The expressive power of these design tools is still limited and there is a necessity to provide the user interface designer with more than one way to design a same user interface providing flexibility of use. Moreover, these have used a simple <name, value> pair to represent context. Processing of context represented using such a simple notation will be more complex and have to make assumptions on the type and sources of context depending on the name in <name, value> pair. To the best of our knowledge, there is no other work that integrated detailed context specification with the context model. A detailed and formal specification, tailored to user interfaces, can be used not only as a basis for context model but also to communicate between the origin of context and the user interface generator.

In order to address the drawbacks associated with suggested context models, we modify the design of task model suggested in [66] to support the contextual specifications of an application. The new resulting context model is called concur context tree. The purpose of this new model is not only to serve the UI designer with increased flexibility but also to provide the application developer with an application protocol interface that aids in the dynamic adaptation of user interface.

3.4.1.   *Design of CCT model*

Two main characteristics that distinguishes Concur Context Tree (CCT) model from the well known concur task tree model are the choice trees and active attribute. The concept of decision nodes has been discussed in [47, 65]. Design of choice tree, is based on the idea of decision nodes, the main difference being the manner in which condition on the choice node is defined. Active attribute when combined with rules specified in rules.xml, allows for specification of effect of context on multiple task nodes spread across the tree, while choice tree allows a localized effect of context.

In addition to the four types of task nodes – abstract, interaction, user and application, we define an additional node called choice node which is associated with a contextual condition. When it evaluates to true the right child of choice node is selected, else the left child is selected. The selected node replaces the choice node. Children may be any type of task node or a choice node. Design of choice task in UML is shown in Figure 3.9
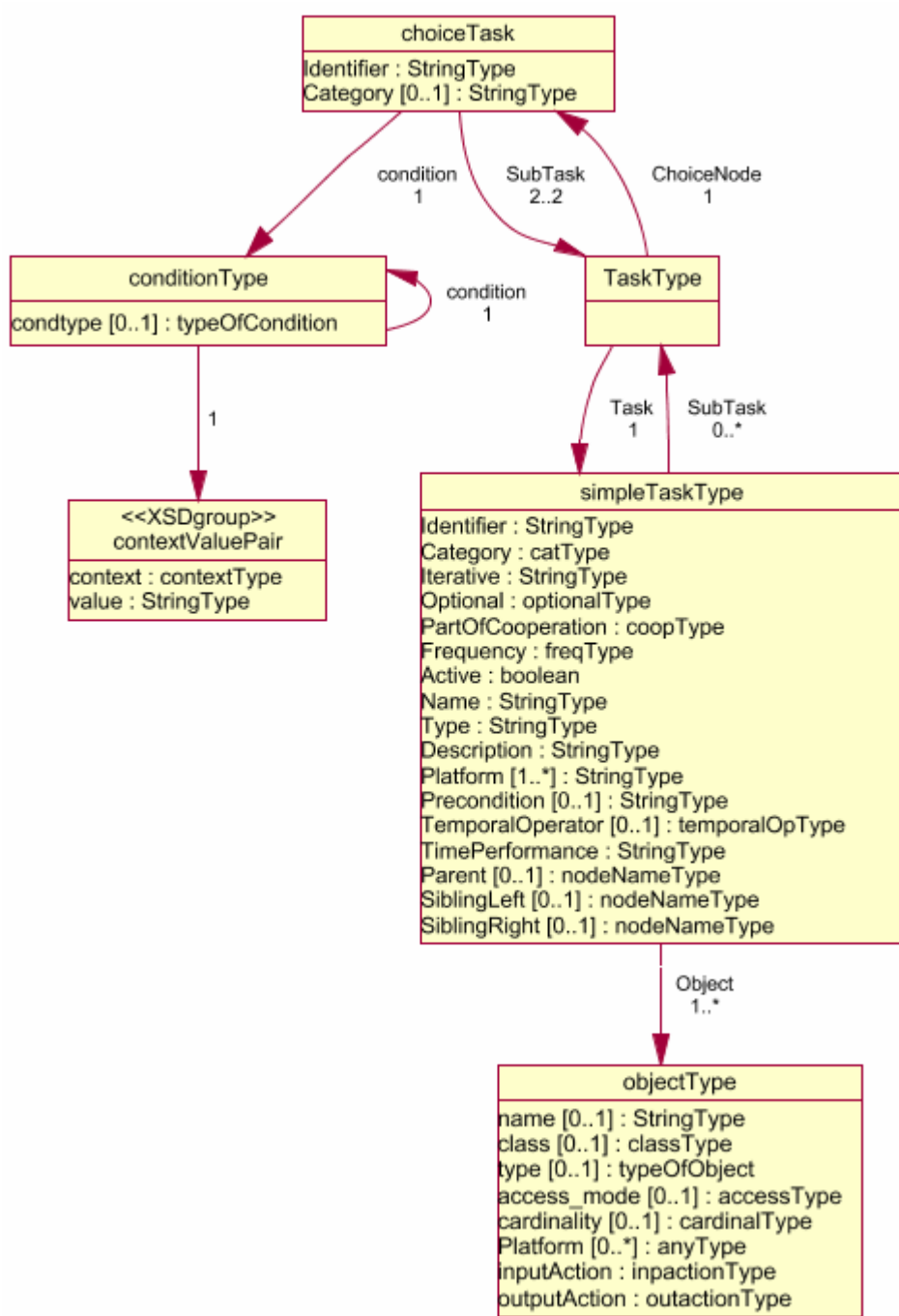
Figure 3.9: Design of Choice Node

We associate an attribute called "active" with each of the task node. The value of this attribute determines if the associated task is active or not. User interface designer may specify conditions that make the task active and those conditions that make it inactive. In cases where a node is made inactive, the task tree is recomputed to reflect the current state of task model. A filtering of the tree based on activity parameter alone is not sufficient to re-compute the task tree. The complication arises with the presence of temporal relationship attribute which has to be recomputed for the neighboring tasks whenever a task node changes its activity state. Hence, to compute a task tree from context tree, events generated from the rule specifications are applied to the tree first followed by filtering or restructuring of the tree. Filtering algorithm has three inputs: node, parameter and value. When a nodes activity parameter is set to false, the node's temporal relationship is checked before removing the node from the context tree while converting it into a task tree. If temporal relationship is one among choice, order independency or concurrent, then, the check succeeds, otherwise the check fails. When the check succeeds, node can be successfully removed from the context tree. In case of failure, an error is reported to the user reporting the inconsistency. Inconsistency arises due to an incorrect rule that asks for removing a node that is needed for other nodes that are not removed. An analogy to this example is a database command that requests for removal of a row in one table whose primary key is being used in an existing row in second table, where second table has a foreign key that is associated with first table's primary key. Hence, our design illustrates the need for reporting errors in rules that violate constraints introduced by temporal relationships.

3.4.2.   Design of Rule specification

Rule specification specifies the actions to be taken when certain contexts defined at the design time are met. Rules specification file contains of one or more rule where each rule comprises of a context condition and one or more actions. In order to know when a rule needs to be executed and when the context needs to be kept track of, rule is associated with its type: Static or dynamic. Identifying a rule as static or dynamic depends on the context and the application requirements. For example, some application requirements may specify that platform context is static where user always uses the same platform while some application requirements may specify that the platform context is dynamic where user might switch between devices and resuming the tasks done on one platform in the other. Context within the rule is specified following the guidelines used to formalize context specification described in earlier section. Context is specified using its plasticity type, context name and one or more attributes, where each attribute is associated with a value. Type of rule is determined by the type of contexts specified in condition. If at least one context is dynamic, then the rule is dynamic, otherwise, rule is static. Figures 3.10 and 3.11 depict the detailed design of rule specification using UML model.
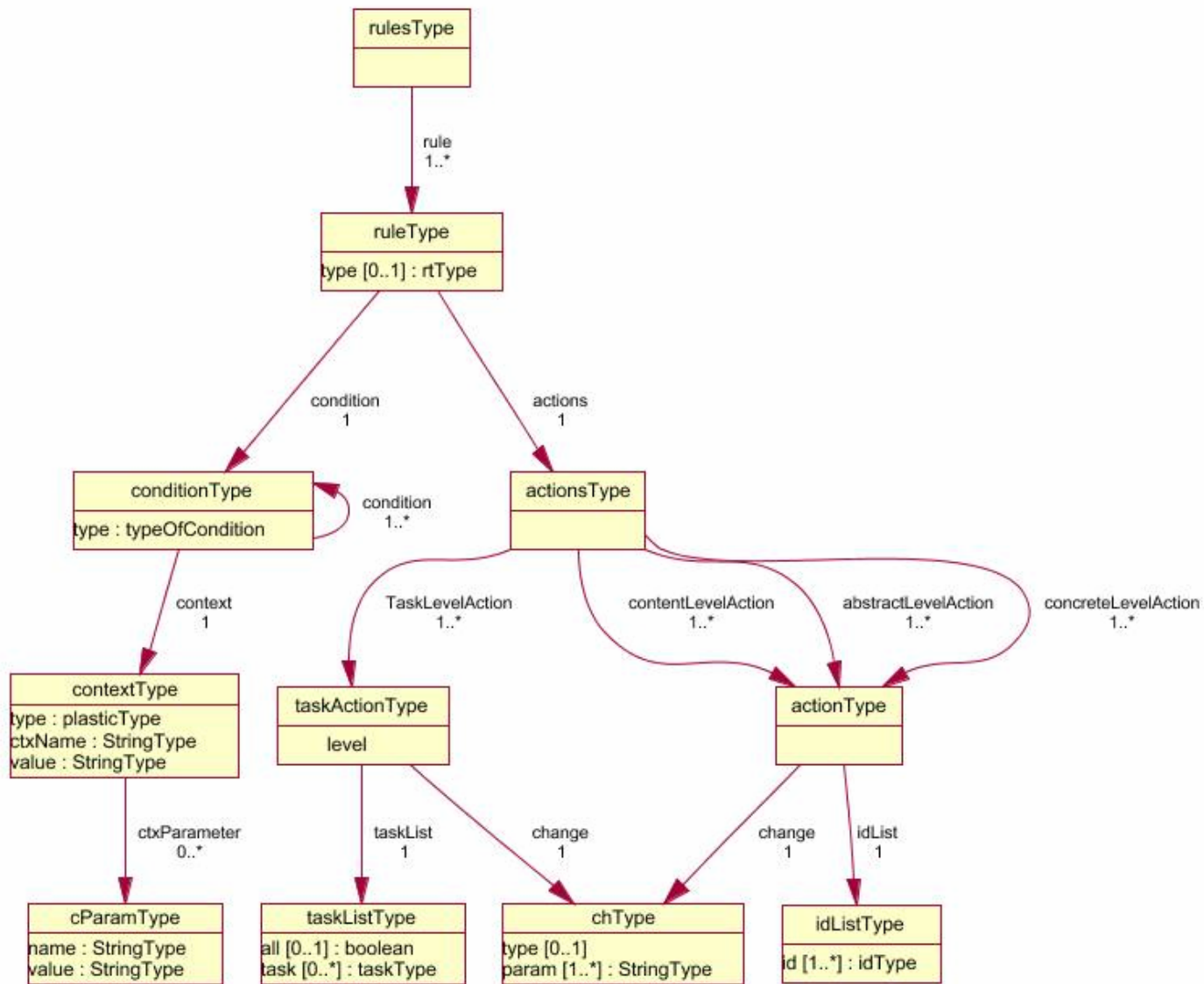
Figure 3.10: Design of rule specification
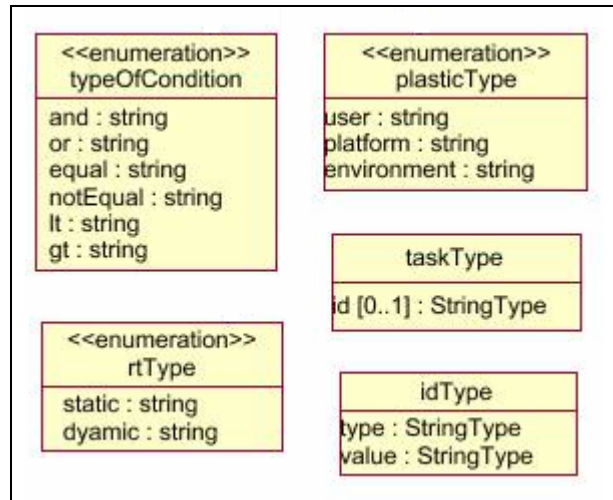
Figure 3.11: Design of rule specification (…continued)

A simple rule shown in Figure 3.12 explains some of the aspects of action specification. Appendix D shows the complete schema design that defines action specification language.

Figure 3.12: Sample rule with condition and actions elements

Specification of actions when a context is met was challenging to design since actions to be taken depend on the level of user interface that is affected. Although, the scope of this thesis is limited to handling contexts, actions and events at task level, the specification has provided a placeholder for propagating the events to other levels as well. The immediate child of 'actions' tag in Figure 3.12 indicates that a context match would perform action at the task level. For actions that are associated with other levels, our design specifies the use of tags such as 'AbstarctLevelAction', and 'ConcreteLevelAction'. Action specifies the change that has to occur. A task level action specifies the changes that need to be made to the components of task

model. Similarly, abstract level actions and concrete level actions specify the changes that need to be made to components of dialogue and presentation model respectively. Hence, each action is associated with components and change to a single or group of components.

Components of models are associated with unique identifiers. For Task model, components are tasks and each task is associated with a unique identifier. An element 'taskList' specifies the list of tasks in the task model. Change is specified using type of change and list of parameters. For task level action, we have identified two different kinds of actions that can be specified: action on individual task by changing the attributes associated with the task and action on group of tasks together which would restructure the task tree. Hence two types of changes have been defined: changeAttribute and restructureTree. Figure 3.13 illustrates the use of these two types of changes. UI designer's task is made easier with informal specifications such as change attribute and restructure tree. Without such a facility, different task trees have to be designed for different contexts.

```
- <actions>
  - <TaskLevelAction>
    - <taskList>
        <task id="Provide request1" />
        <task id="Provide request2" />
        <task id="Enter Zip Code" />
        <task id="Provide request3" />
      </taskList>
    - <change type="changeAttribute">
        <param>Active</param>
        <param>false</param>
      </change>
    </TaskLevelAction>
  - <TaskLevelAction>
    - <taskList>
        <task id="currentTask" />
      </taskList>
    - <change type="restructureTree">
        <param>addTask</param>
        <param>child</param>
        <param>choice</param>
        <param>confirmOrChangeLocation</param>
      </change>
    </TaskLevelAction>
  </actions>
```

Figure 3.13: Illustrating changeAttribute and restructureTree

One of the two types of changes 'changeAttribute' is used to specify that an attribute of a task or its object may be changed to a specific value. It is associated with two or more parameters. With two parameters, first parameter specifies the name of the attribute and the second

specifies the value it has to be set to. Examples of attribute names include 'active', 'platform', 'category', 'type', 'name'. Attribute 'name' is associated with both task and all of its objects. In cases such as these, first parameter explicitly specifies the identifier of the object if change needs to be done to the object instead of a task.

Second type of change 'restructureTree' specifies action to be taken on group of components together. It is associated with two or more parameters where the first parameter always specifies the type of restructuring. Parameters that follow specify the information necessary to perform the type of restructuring specified in first parameter. We have identified two types of restructuring: sortSiblings, and addTask. 'sortSiblings' sorts the siblings of the task, specified in 'taskList', based on an attribute, which is specified as a second parameter in the 'change' tag. This may be applied only if the siblings are temporally related to each other with a choice operator. 'addTask' adds a task tree as a child or sibling of the task specified in taskList. Second parameter specifies weather it is a child or left sibling or right sibling. Third parameter specifies temporal relationship between root of the task tree to be added and the task node specified in 'taskList'. Last parameter specifies name of a method which returns the task tree that has to be added. Additional types of restructuring may be added as desired.

3.5   CONCLUSION

We have presented in this chapter our proposed framework for achieving context aware and adaptive user interface generation. Data flow between different components has been explained. Since the scope of the thesis is limited to transiting from context requirements to task model, our specifications and models contribute only to a part of the framework.

One of the advantages of using a task tree is that it is convenient for the designer to think and design in terms of hierarchical manner. A graphical tool that can help the designer to specify the operations on this tree will make it more useful. This graphical tool, called 'context modeling tool' is yet to be developed and is one of the entries into future work.

CHAPTER 4: SAMPLE APPLICATION

We present in this section a sample application illustrating how the proposed framework can be used to dynamically change the user interface to changing contexts.

4.1   ADAPTATION OF USER INTERFACE FOR PROVIDING SERVICES TO SENIOR CITIZENS

Problem Specification: A web application designed to provide services to senior citizens is often accessed by healthy care takers of senior citizens or senior citizens themselves. Web site has to be designed for numerous contexts including various types of users accessing the application in different environments with different devices. With senior citizen accessing services through a web application, user interface of the application needs to adapt to user's static and dynamic characteristics. Context aware user interface requirement of the application can be stated as "Adapt the user interface according to user's context of use comprising of user's static and dynamic profile, platform and environment."

Examples where adaptation is needed: If the user has an Alzheimer's disease, user interface has to be adapted to indicate current position of the user with respect to home page in order to remind the user of current and past activity. When a user's usage characteristics related to mouse exceeds the recommendation given by the doctor, user interface has to be adapted to alert the user and facilitate him to change the mode of interaction. In this example, the context is mouse movement. In order to reduce the amount of interaction using a standard keyboard or mouse input and screen output, there needs to be a facility for the UI facets such as layout and navigation to adapt to the frequency of usage. Adaptation of layout may include order of links, distance between links, replacing links with images, and increasing the font. User's static preferences may specify that user prefers to interact with graphical items rather than textual items.

Next, we present the task model and action specification defined for this application to illustrate how user interfaces adapt to change in the contexts. The work presented in this thesis has been used to generate task models from context specification. For illustrating the obtained results, we have used the Teresa tool to generate the final user interface form task model. For generation of presentation task sets (PTS), we have used the open source software Tasklib that generates PTS from ConcurTaskTree model.

To design and develop a user interface for any application, our framework requires the following tasks to be done:
1. From the requirements of the application, design a context model.
2. Identify the list of contextual parameters that the application need to support.
3. Specify rules. Rule, as explained in chapter 3, comprises of context condition and actions to be taken when the condition is met.

Actions specified in the rules can comprise of call to methods that are application specific. Our framework provides a place to support such application specific methods.

User interface developer implements application specific methods, if any, identified at the design time. Tools may be used to specify actions and adaptation methods at dialogue and presentation level also.

As a proof of concept to our framework, we present in this section context parameters, context model and five rules that adapt context model and hence adapt the user interface. Since this is only a proof of concept for demonstration purposes, we have used minimal set of requirements and rules for the complicated application. User interactions and contextual conditions are specified using context model, design of which is described in section 4.1.2. Context aware adaptation requirements of this application are specified using context and rule specifications described in sections 4.1.3 and 4.1.4.

4.1.1.   *Design of context model*

Figure 4.1 depicts part of the context model for modeling services to senior citizens. After login, user will be able to access one or more of six services: Transportation, home repairs, access health information, order meals, insurance counseling, and legal and ethical issues. One of the six types of services, 'order meals' is elaborated in the figure. For order meals service, user will be able to enter the zip code if the system is not already aware of it, and request for available services associated with 'order meals'. When user selects one of the system displayed services, he can confirm the address, proceed to order and continue using the web application for other services.

Application requirements specify that, if location of user is known for sure, user is not prompted to enter zip code information. Hence the task 'Enter Zip code' may or may not be active in the resulting task model. UI designer can specify reflect this requirement in his design in one of the following two ways:
1. Specify this condition within the context model by using a choice node which specifies the condition that
2. Specify the condition as a part of rule in rule specification.

For sake of demonstration, we chose option 2 in the design that follows.

Context model
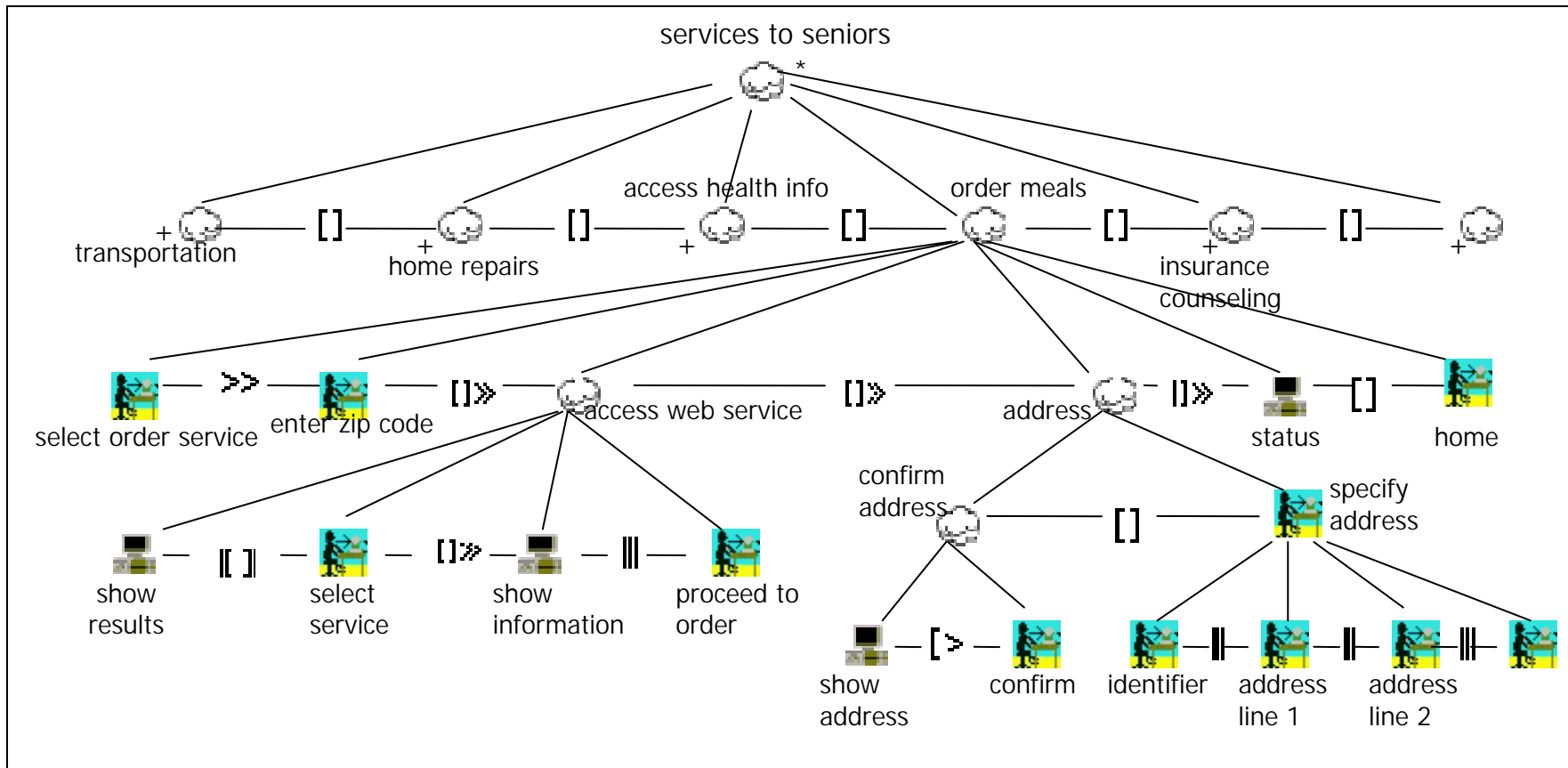
Figure 4.1 Context model for services provided to senior citizens

4.1.2.   *Identification of contextual parameters*

Again the application requirements are used to identify contextual parameters. During the design time, for each identified contextual parameter, the type and nature of the parameter is also specified. For the current application 'services to senior citizens', we demonstrate the adaptation capabilities using the following contextual parameters:
1. Type of navigation label. Values: graphical or textual links
2. Frequency of accessing a service
3. Intensity of mouse movement
4. Time and activity after login
5. Device type. Values: desktop or PDA
6. Location

In order to fit into our proposed framework, UI designer is required to specify plasticity type, level and nature of contextual parameter. For example, application requirements may either specify that device type is a static context where any device may be used to access the application but, once a device is used, the user would not switch between devices during interaction with the application. In this case, device type is static. In certain applications changing device type may be a requirements in which case the context maps to dynamic context. Hence, depending upon the application requirements the designer specifies the nature as well as the level of interface that is effected.

	Plasticity	Context name	Nature	Level
Type of navigation label.	User	User preference	Static	Presentation
Frequency of accessing a service	User	User behavior	Dynamic	Task
Intensity of mouse movement	User	User behavior	Dynamic	Task
Time after login	User	User activity	Dynamic	Task
Activity after login	User	User activity	Dynamic	Task
Device	Platform	Device type	Dynamic	Task, concrete
Location	Environment	Location	Dynamic	Task

Table 9: Context specification

In order to represent the context requirements, first step performed by the user interface designer is to analyze the context specific application requirements and specify them. Result of such analysis for the sample application in shown in Table 9.

4.1.3.   Rule Specification

Rules are made of context conditions and actions. For the current case study, we identified five conditions, and hence five rules. Each condition, either simple or complex, is made of one or more of the contextual parameters identified in the section 4.1.2. Each condition is associated with one or more actions. Possible actions to be taken are constrained by level information given during context specification in Table 10. Following are list of five rules considered for demonstration of our framework.

1. If user prefers to navigate using graphical links instead of textual links, a graphical link is displayed instead of textual link
2. If platform is changed to PDA, change ordering to numbering instead of bullets and change navigation type to textual links.
3. If frequency of any of accessing any of the six services provided to user increases by a value of five, re-order links to access services in decreasing order of frequency.
4. If user behavior specifies that he should not use system for long time, then monitor for mouse movement. If motion of mouse increases a threshold value, then user is suggested to switch type of interaction.
5. If location of user is known in advance, then do not prompt the user to enter zip code.

These informal specifications can be given as an input to the context modeling tool to generate a formal specification action.xml given in appendix F.

4.1.4.   Results

With the model shown in Figure 4.1 as input to Teresa tool, list of 15 presentation XHTML pages are created. Teresa tool provides a facility to either generate final user interface directly from task model or let the user intervene in this UI generation. User interface generation can be controlled at abstract and concrete user interface levels. For generating the final user interface, we have intervened during the UI generation, instead of allowing the Teresa tool to automatically generate the UI. The purpose of user involvement during the generation is to change the background color, and titles associated with each screen. Same set of operations are done at the presentation level each time final user interface is generated from the task model.

Figure 4.2 depicts initial user interface generated when no rule is applied. The six services provided to the users are represented as textual hyperlinks.

Figure 4.2: User interface generated with default context

Rule 1: *If user prefers to navigate using graphical links instead of textual links, a graphical link is displayed instead of textual link.*

Rule specification: Formal specification of each of the five rules for this case study is specified in Appendix F.
Condition: user's current platform is Desktop and user's preference for navigator choice is graphical link
Concrete level action: Perform a concrete level action for the six identifiers identifying each of the six tasks

Context specification of Table 9 specifies that this context effects concrete level. The effect of changing the textual links to graphical links is made possible by changing the parameters at the concrete user interface. Here the intermediate stage of Teresa output gives the option of changing parameters at concrete level. Modification has been done using Teresa tool to concrete UI in order to produce the result shown in Figure 4.3. The main purpose of this example is to illustrate the capability of proposed rule specification to specify concrete level actions. Remaining rules that follow illustrate in detail, the support given by the rule specification to handle task level actions.

Figure 4.3: User interface generated rule 1 is applied

Rule 2: *If platform is changed to PDA, change ordering to numbering instead of bullets and change navigation type to textual links*

Condition: user's current platform is PDA
Task level action: change 'platform' attribute of all tasks to PDA
Rule 2 generates a user interface appropriate for a PDA (Figure 4.4):
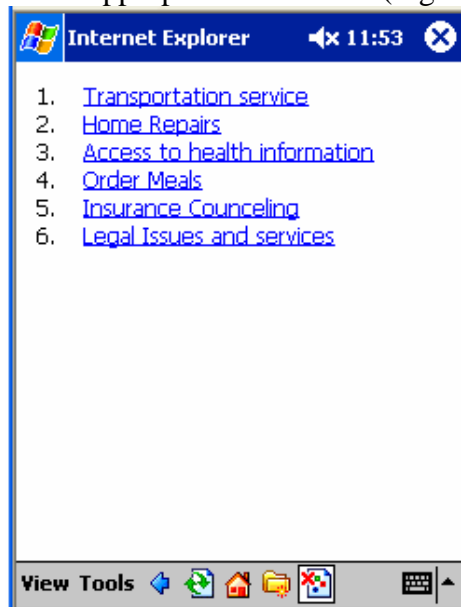
Figure 4.4: User interface generated when rule 2 is applied

'changeAttribute' method defined in the proposed rule specification is used to specify the action part of this rule. This rule uses two parameters to changeAttribute: first one, platform, indicates the attribute name; Second, PDA, indicates value that the attribute will be assigned after rule is applied.

Rule 3: *If frequency of any of accessing any of the six services provided to user increases by a value of five, re-order links to access services in decreasing order of frequency.*

Condition: task frequency increase of one of the six tasks is greater than 5
Task level action: restructure tree to sort siblings based on decreasing order of frequency.

'restructureTree' method defined in the rule specification to support task level actions is used in this rule. This rule changes the order of links as depicted in Figure 4.5.
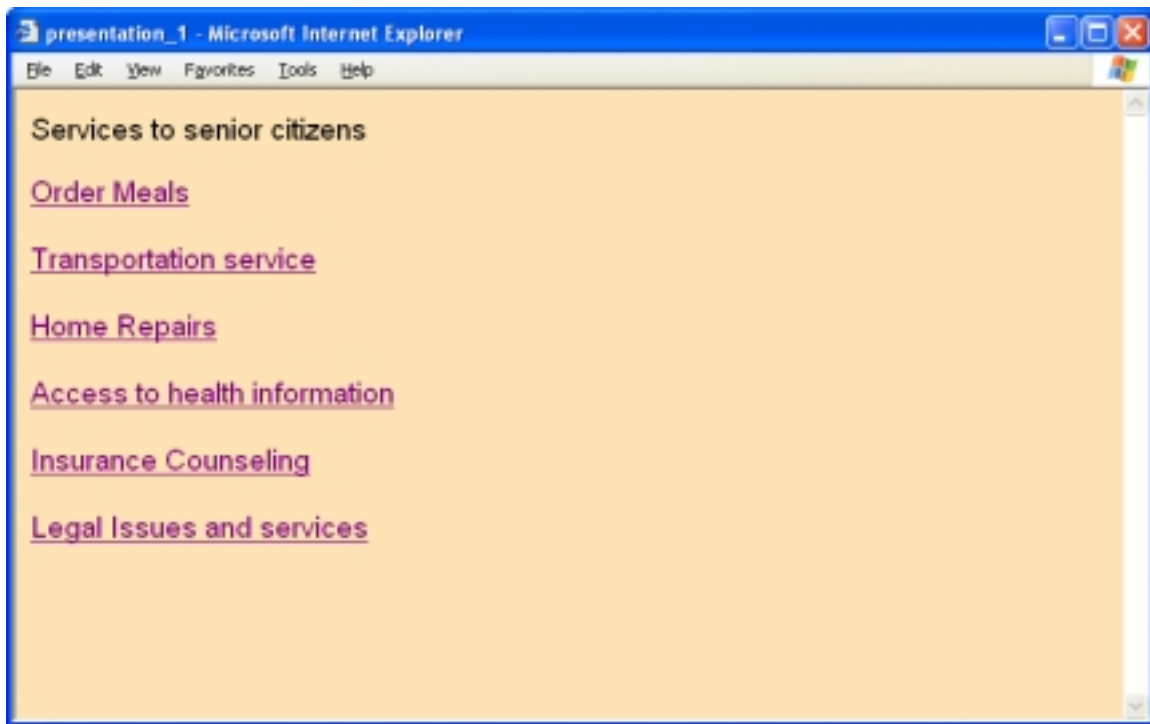
Figure 4.5: User interface generated when rule 3 is applied

Rule 4: *If user behavior specifies that the user should not use system for long time, then monitor for mouse movement. If motion of mouse increases a threshold value, then user is suggested to switch type of interaction.*

Condition: Mouse movement parameter of user behavior is greater than 300 units
Action: Restructure tree to add a new task as a child to the current task, where new task alerts the user about his behavior.

Before the context is met	After the context is met
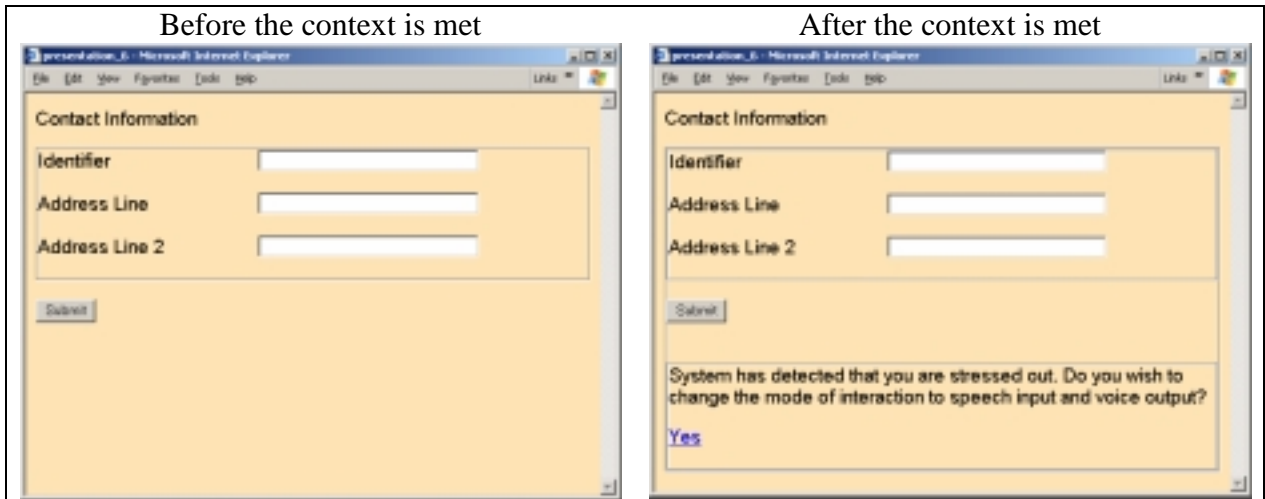
Figure 4.6: User interface generated before and after rule 4 is applied

Figure 4.6 depicts the screen before and after rule 4 is applied. Again, 'restructureTree' method defined in rule specification is used to perform the action for this rule. The action, shown in Appendix F, contains four parameters. First one specifies the type of restructuring, which in this case is 'addTask'. The next parameters that follow specify weather task is added as a child, right sibling or left sibling; the temporal relationship, and the method that returns root of the new task. In this case, second parameter is child, operator is choice and method is 'alertExtensiveMouseMotion'

Rule 5: If location of user is known in advance, then do not prompt the user to enter zip code.

Condition: Both x and y co-ordinate values of location parameter are not null
Action: change 'active' attribute of those tasks that require zip code entry to false
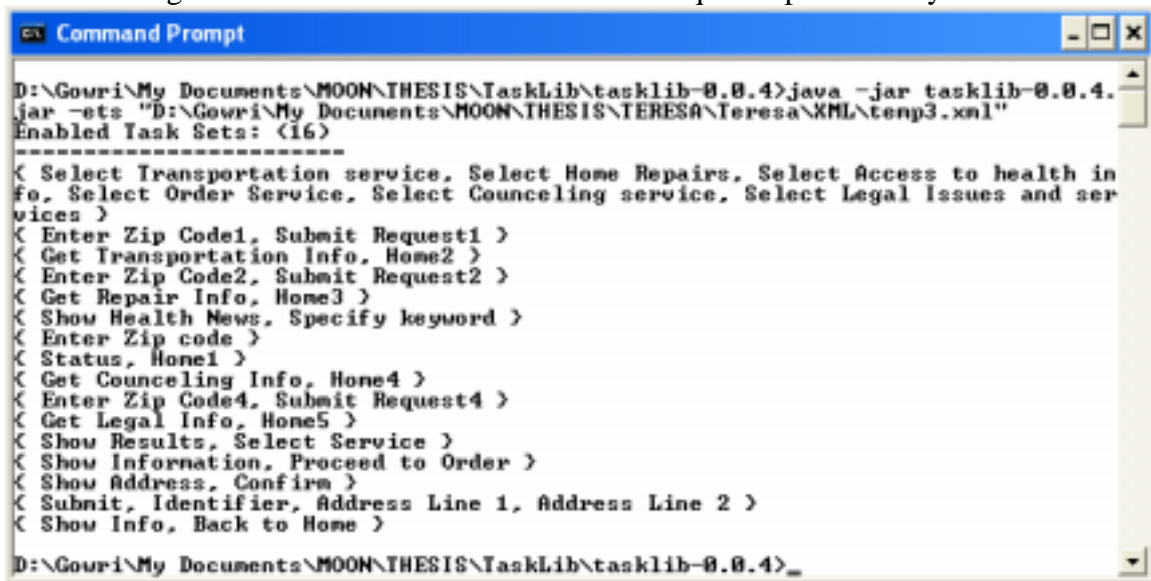
Figure 4.7: Presentation Task sets calculated before rule 5 is applied

64

Active parameter is set to false using 'changeAttribute' method of rule specification. Recollecting the flow of information in the proposed framework, context server triggers an event indicating the action to be taken when the condition specified in this rule is met. Actual business logic that transforms context tree to task tree is in the interface and modeling server. Whenever the attribute to be changed is active parameter, the context model is filtered against this 'active' parameter to produce the task tree.

Application of this rule to the context model produces a task tree with lesser number of enabled task sets (ETS). Section 2.3.3.2 in this document explains definitions of ETS and PTS. This rule changes the navigation between the screens. We have used the open source library TaskLib to take the new task tree and generate the list of enabled task sets. The generated enabled task sets are shown in Figure 4.7 and 4.8 with Figure 4.6 depicting ETSs when location is not known and Figure 4.8 depicting ETSs when location is known.

Figure 4.8: Presentation Task sets calculated after rule 5 is applied

PTS1. Selecting Order meals will enable PTS2

PTS2. When Zip Code is entered and button pressed, PTS3 is enabled

PTS3
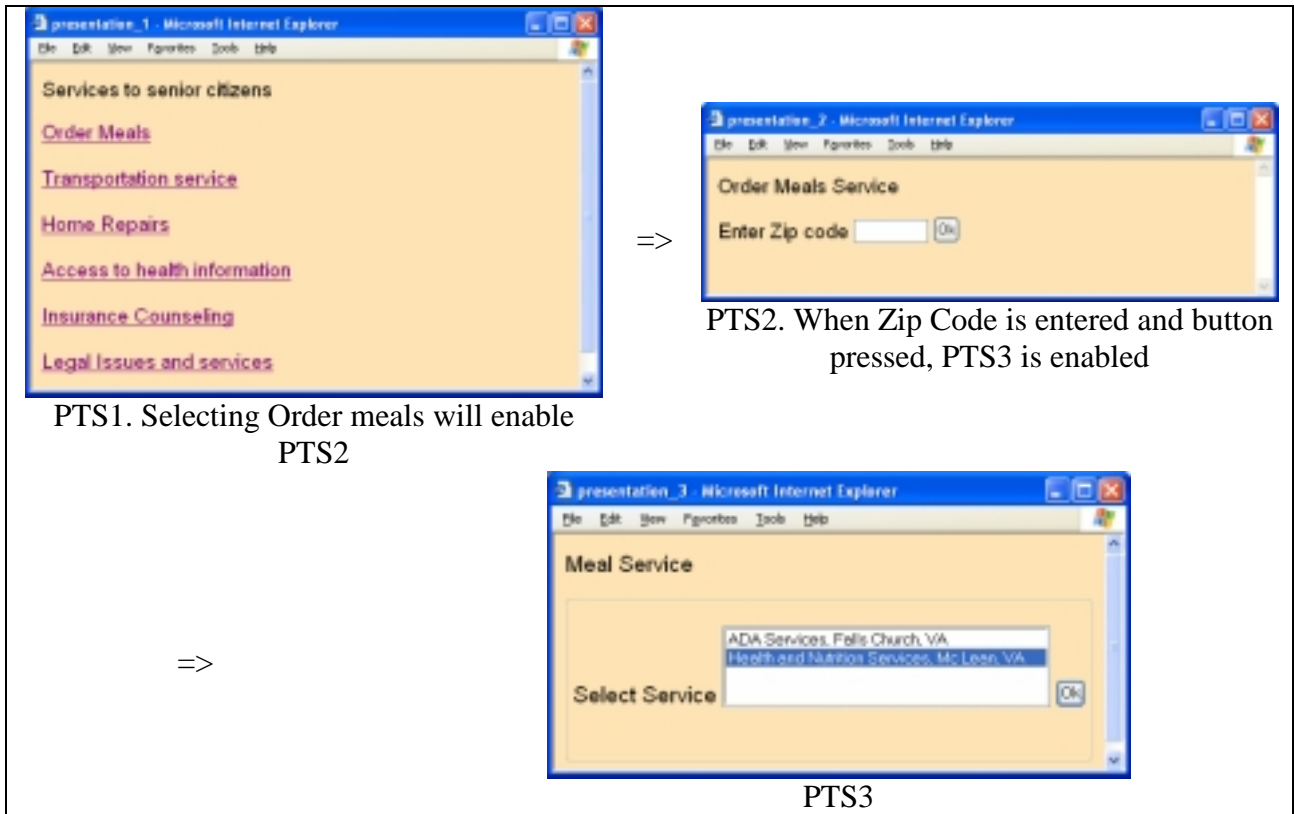
Figure 4.9: Navigation sequence before context in rule 5 is met

Figure 4.9 illustrates the navigation when location is unknown. When location is known, user skips the intermediary step that requests for location information. With no heuristics applied, number of presentation task sets equals the number of enabled task sets. Hence, the number of presentation screens is reduced after the context specified in rule 5 is met.

To summarize, the example application illustrates how our framework is used to specify context (Table 9), context model (Figure 4.1) and rule specification (Appendix F) in order to achieve context aware and adaptive user interface generation. Context Specification (Table 9) shows use of level, nature and plasticity type information. With our proposed rule specification, rule 1 illustrated the support for specifying concrete level actions. Rule 2 illustrated the use of change attribute method. Rule 3 and 4 illustrated use of restructure tree method with different type of restructuring: sort siblings and add task. Rule 5 illustrated use of 'active' attribute and filtering based on 'active' attribute.

CHAPTER 5: CONCLUSIONS AND FUTURE WORK

The contributions outlined in Chapter 1 have been met by the design and proof of concept implementation that validates the design of context aware and adaptive user interface generation. The thesis illustrates the need for formal context specification, flexible and extensible way of specifying context model and integration of the two.

Context modeling tool can be used by user interface designer to specify context and context model, and the tool would generate the formal context specifications and context model in form of xml. Although we have mentioned that this tool could be used, we have not yet developed this tool and the current work started from the context specification and context model in form of xml. However, it is highly essential to have a visualization environment to support the work presented in this thesis. This remains as future work. Algorithm designed for filtering based on constraints introduced by temporal relationships is done; but is yet to be implemented. Although we have defined the context model with choice node, we have not yet illustrated its use in the scenario presented in chapter 4.

The research issues associated with user interface modeling and generation are many-fold. Researchers have been actively involved in addressing various research issues associated with this goal. With the limited scope, we have contributed to part of the current research activity. The applications that can benefit from this research are also many-fold.

5.1   OPEN ISSUES

User interface design is not the goal of UML although UML could be used for designing user interfaces. However, the goal of task model is to design user interfaces for interactive applications. Hence, task model is easier to use for that purpose. However, there is currently no agreement on how a task should be described and represented adequately and completely. There are no standards for specifying a task model. Task analysis is still a research issue and research community needs to work towards a robust technique and standard for task representation. Similarly, there are currently no standards for specifying context model, dialogue model and presentation model.

Transition from each model to the consecutive model is a research challenge in itself. User interface generation life cycle in model based UI design and development involves transition from one model to the consecutive model. It is also challenging to balance the amount of automation and manual process involved in transition between consecutive models. Context model to task model transition has been presented in this thesis. Automation of task model to dialogue model transition and dialogue model to presentation model are also currently faced with challenges, description of which is beyond the scope of this thesis.

Context aware frameworks form an essential part of adaptive user interface generation. We see context toolkit as a promising framework due to its distributed and loosely

coupled nature. Java context aware framework has similar features. However, it is intended for domain specific and research purposes, not yet well established, less documented as of now and might show similar strengths as context toolkit as it is evolving. Both these frameworks have issues with security features. Among other issues, as described in chapter 2, Context toolkit provides lower level programming abstraction. Application developers are burdened to discover the context widgets, context interpreters, context aggregators, and discoverers. Just like user interfaces was not the goal of UML, similarly user interfaces is not the goal of these context aware frameworks. Hence, there is highly a need for building a context toolkit that is tailored towards adaptive user interfaces. The current thesis does provide the characteristics of event generator and context representation for processed context. Work has to be done to integrate these with other components of context framework. The result of such integration would be a unique toolkit that propagates events to appropriate modeling level of user interface generation lifecycle. There is currently no toolkit for context abstraction and processing that was built for user interface generation.

There is also a need to develop adaptive user interface toolkits for applications that require context aware and adaptive user interface generation. Context toolkit described above will be a part of this larger toolkit. There is currently no such context aware and adaptive user interface toolkit that can readily be used by the user interface designers. Framework introduced in this thesis shows all the components that are necessary for such a toolkit. It illustrates the use of context handling toolkits, modeling including task, context, dialogue and presentation modeling, and use of XML based user interface description languages for representing the results from modeling. As described earlier, building a context aware and adaptive user interface toolkit would involve addressing the numerous issues related to context awareness, model representations, modeling algorithms, and user interface description languages. We believe that it is highly essential to have a coordinated and integrated effort from the researchers across the world to achieve this goal.

REFERENCES

[1]   "Skyscape PDA Usage Survey," Available at http://www.palminfocenter.com/
palm/p_story.asp?ID=6357, December, 2003

[2]   Knag, Paul "HIPAA: a guide to health care privacy and security law", Aspen
Publishers, Inc, January 2002

[3]   Scott Mirtchell, Mark D. Spiteri, John Bates, George Coulouris, "Context-aware
multimedia computing framework in the intelligent hospital. ACM SIGOPS
European Workshop 2000: 13-18

[4]   Hui Lei, Daby M. Sow, John S. Davis II, Guruduth Banavar, Maria R. Ebling
"The design and applications of a context service" Mobile Computing
Communications Review, October 2002, Volume 6, Number 4, pp 45-55

[5]   Miguel A. Muñoz, Marcela Rodríguez, Jesus Favela, Ana I. Martinez-Garcia,
Victor M. González , "Context aware mobile communication in hospitals" IEEE
Computer. September 2003, Volume 36, Number 8, pp. 60-67.

[6]   Marcela   Rodríguez,   Jesus   Favela.   "Autonomous   Agents   to   Support
Interoperability and Physical Integration in Pervasive Environments". Atlantic
Web Intelligence Conference, AWIC 2003, Springer-Verlag, LNAI 2663, Madrid,
Spain, May 2003. pp.278-287

[7]   Rodríguez, M., P. Arroyo, A. I. Martínez, J. Favela and C. Navarro. "Autonomous
Agents as Conference Aids in Ubiquitous Collaborative Environments", In
Proceedings of the 2003 International Symposium On Collaborative Technologies
and Systems (CTS'03), Orlando, Florida, USA. pp 19-24

[8]   Anind K. Dey "Understanding and Using Context" Personal and ubiquitous
computing, Volume 5, February 2001, pp 4-7

[9]   Anind K. Dey and Gregory D. Abowd "Towards a Better Understanding of
Context and Context-Awareness" In Proceedings of CHIA'00 workshop on
Context-Awareness, 2000.

[10] Anind K. Dey, Daniel Salber, Gregory D. Abowd, Masayasu Futakawa "The
Conference Assistant: Combining Context Awareness with Wearable Computing"
3rd International Symposium on Wearable Computers, October 18 - 19, 1999, San
Francisco, California, pp. 21

[11] Patrik Floreen, Greger Linden "An introduction to context aware computing"
Available at: http://www.cs.helsinki.fi/hiit_bru/courses/582446/, October 2003

[12] Brown, P., Burleston, W., Lamming, M., Rahlff, O., Romano, G., Scholtz, J., and
Snowdon, D., "Context- Awareness: Some Compelling Applications,"
Proceedings the CH12000 Workshop on The What, Who, Where, When, Why
and How of Context-Awareness, April 2000.

[13] Glenn Judd, Peter Steenkiste , "Providing contextual information to pervasive
computing applications" First IEEE International Conference on Pervasive

Computing and Communications (PerCom'03) March 23 - 26, 2003 Fort Worth, Texas

[14] David Garlan, Daniel P. Siewiorek, Asim Smailagic, and Peter Steenkiste "Project Aura: Towards Distraction Free Pervasive computing" IEEE Pervasive Computing, special issue on "Interated Pervasive Computing Environments", Volume 1, Number 2, April-June 2002, pages 22-31

[15] Henrik Bærbak Christensen , Jakob Bardram, Supporting Human Activities - Exploring Activity-Centered Computing, Proceedings of the 4th international conference on Ubiquitous Computing, p.107-116, September 29-October 01, 2002, Göteborg, Sweden

[16] Jakob E. Bardram and Henrik Baerbak Christensen "An initial comparison between Aura and Activity Based Computing" Report, March 7, 2003

[17] J. E. Bardram. "Activity-Based Support for Mobility and Collaboration in Ubiquitous Computing." In L. Baresi, editor, Proceedings of the Second International Conference on Ubiquitous Mobile Information and Collaboration Systems, Lecture Notes in Computer Science, pages 101–115, Riga, Latvia, Sept. 2004. Springer Verlag.

[18] J. E. Bardram, R. E. Kjær, and M. Pedersen. Context-Aware User Authentication – Supporting Proximity-Based Login in Pervasive Computing. In A. Dey, J. McCarthy, and A. Schmidt, editors, Proceedings of Ubicomp 2003: Ubiquitous Computing, volume 2864 of Lecture Notes in Computer Science, pages 107–123, Seattle, Washington, USA, Oct. 2003. Springer Verlag

[19] Joao Pedro Sousa and David Garlan "Aura: An architectural framework for user mobility in ubiquitous computing environments" in Software Architecture: System Design, Development, and Maintenance (Proceedings of the 3rd Working IEEE/IFIP Conference on Software Architecture) pp 29-43, August 2002.

[20] J. E. Bardram, C. Bossen, A. Lykke-Olesen, K. H. Madsen, and R. Nielsen. Virtual Video Prototyping of Pervasive Healthcare Systems. In Conference proceedings on Designing Interactive Systems: processes, practices, methods, and techniques (DIS2002), pages 167–177. ACM Press, 2002.

[21] Jakob. E. Bardram. Hospitals of the Future – Ubiquitous Computing support for Medical Work in Hospitals. In J. E. Bardram, I. Korhonen, A. Mihailidis, and D. Wan, editors, UbiHealth 2003: The 2nd International Workshop on Ubiquitous Computing for Pervasive Healthcare Applications.

[22] Jakob E. Bardram "Applications of ContextAware Computing in Hospital Work – Examples and Design principles" ACM, SAC '04, March 14-17, 2004, Nicosia, Cyprus

[23] Jakob Langdal, Balasuthas Sundararajah, Kenneth-Daniel Nielsenkenneth "Context aware automatic home automation system: CAHAS January 14, 2004" Aarhus University, Draft, January 14, 2004

[24] Jeffrey Hightower, Gaetano Borriello "Location systems for ubiquitous computing" August 2001 (Vol. 34, No. 8) pp. 57-66

[25] J. E. Bardram and H.B. Christensen. "Middleware for Pervasive Healthcare", A White Paper. In Workshop on Middleware for Mobile Computing, Heidelberg, Germany, 2001.

[26] Kimmo Raatikainen, Henrik Bærbak Christensen, Tatsuo Nakajima "Application requirements for middleware for mobile and pervasive systems" Volume 6 , Issue 4 (October 2002) Pages: 16 - 24

[27] Peter J. Brown, John D. Bovey, and Xian Chen. Context aware applications: from the laboratory to the marketplace. IEEE Personal Communications, 4(5):58-64, October 1997.

[28] "UbiHealth 2003: The 2nd International Workshop on Ubiquitous Computing for Pervasive Healthcare Applications, Seattle, Washington, October 12, part of the UbiComp 2003 Conference." http://www.healthcare.pervasive.dk/ubicomp2003/papers/

[29] "Mobile computing in a hospital: the WARD-IN-HAND project" ACM March 19-21, 2000 http://www.wardinhand.org/

[30] Richard W. DeVaul, Alex "Sandy" Pentland "The Ektara Architecture: The Right Framework for Context-Aware Wearable and Ubiquitous Computing Applications." February 2000

[31] "The Java Context Awareness Framework (JCAF) – A Service Infrastructure and Programming Framework for Context Aware Applications." Available at http://www.pervasive.dk/publications/files/bardram_jcaf_2004.pdf, February 2004

[32] Project page of Activity based Computing. Available: http://www.daimi.au.dk/~bardram/abc/index.php, Accessed in January 2004

[33] Gross, T. & M. Specht (2001). Awareness in Context-Aware Information Systems. In Oberquelle H., R. Oppermann & J. Krause (Eds.) Mensch & Computer - 1. Fachübergreifende Konferenz. Bad Honnef (Germany). 173-182.

[34] Schilit, B., Adams, N. Want, R. Context Aware Computing Applications. 1st International Workshop on mobile computing systems and applications (1994) 85-90

[35] G. Chen and D. Kotz, A survey of context-aware mobile computing research Technical Report TR 2000-381, Department of Computer Science, Dartmouth College, November 2000

[36] Rhodes, B. & Maes, P. (2000). Just-in-time information retrieval agents. IBM Systems Journal 39(3-4), pp. 685-704.

[37] J. Pascoe, N. Ryan and D. Morse, "Issues in developing context-aware computing," in Proc. Intl. Symposium on Handheld and Ubiquitous Computing, LNCS 1707, Springer, 1999, 208-221

[38] Pascoe, J. Adding Generic Contextual Capabilities to Wearable Computers. 2nd International Symposium on wearable computers (1998) 92-99

[39] "A Survey of context awareness" Available from: http://www.comp.lancs.ac.uk/ ~km/papers/ContextAwarenessSurvey.pdf Accesssed in January 2004

[40] An Integrated Contextual Information Service for Pervasive Computing Applications Glenn Judd and Peter Steenkiste January 2003 Abridged version published in the Proceedings of the First IEEE International Conference on Pervasive Computing and Communications. Dallas-Fort Worth, TX. March 2003. Src: http://reports-archive.adm.cs.cmu.edu/anon/2003/CMU-CS-03-100.pdf

[41] Composite Capabilities/ Preference Profiles (CC/PP) information page. A W3C standard available at http://www.w3.org/Mobile/CCPP/, Accessed in April 2004

[42] Calvary, G., Coutaz, J., Thevenin, D.: A Unifying Reference Framework for the Development of Plastic User Interfaces. In: Proc. of IFIP WG 2.7 Conf. on Engineering the User Interface EHCI'2001 (Toronto, May 11-13, 2001). Chapman & Hall, London (2001).

[43] Workshop on XML based High level User Interface Descriptions. Developing User interfaces with XML: Advances on User Interface Description Language. 25th May 2004. Selected papers available: http://www.edm.luc.ac.be/uixml2004/

[44] Ahmed Seffah, Homa Javahery "Multiple User Interfaces, cross platform Applications and context aware interfaces" John Wiley & Sons, Ltd ISBN: 0470854448, January 2004

[45] Thevenin, D., Coutaz, J.: Plasticity of User Interfaces: Framework and Research Agenda. In: Sasse, A., Johnson, Ch. (eds.): Proc. of IFIP TC 13 Int. Conf. on Human-Computer Interaction Interact'99 (Edinburgh, August 1999). IOS Press, London (1999) 110–117

[46] "XML Markup Languages for User Interface Definition" Available: http://xml.coverpages.org/userInterfaceXML.html, Accessed in March 2004

[47] June 2001 "Task Modelling for context-sensitive user interfaces" By Costin Pribeanu, Quentin Limbourg, and Jean Vanderdonckt. In Proceedings of the Eight Workshop of Design, Specification and Verification of Interactive Systems, June 13-15, 2001

[48] Manuel A. Perez-Quinones, comment on "Academic Perspective on UIML" http://lists.oasis-open.org. Accessed in March 2004

[49] J. Eistenstein, J. Vanderdonckt, and A. Puerta. Applying model-based techniques to the development of user interfaces for mobile computers. In Proc. ACm Conference on Intelligent User Interfaces IUI' 2001, pp 69-76

[50] "HCI Issues in eXtreme Computing"   http://endeavour.cs.berkeley.edu/talks/ LandayUI.ppt

[51] Qiyang Chen, "Human Computer Interaction: Issues and Challenges." Idea Group Publishing, 16 March 2001, ISBN: 1878289918

[52] Angel R. Puerta and David Maulsby "MOBI-D: A Model-Based Development Environment for User-Centered Design" CHI '97, Atlanta, March 1997, pp.4-5

[53] D. Sinnig, P. Forbrig, and A. Seffah, "Patterns in Model-Based Development", Position Paper in INTERACT 03 Workshop entitled: Software and Usability Cross-Pollination: The Role of Usability Patterns, September 2003.

[54] B. Myers, S. E. Hudson, R. Pausch, "Past, Present and Future of User Interface Software Tools" ACM TOCHI V7, N1, March 2000, pp 3-28.

[55] Joe Tullio "Model-based UI Tools" http://www.cc.gatech.edu/classes/AY2003/ cs4470_fall/lectures/lecture36-model-based.ppt, CC6456 November 2002.

[56] Q. Limbourg, J. Vanderdonckt, B. Michotte, L. Bouillon, M. Florins & D. Travisan, CXML : A User Interface Description Language for Context-Sensitive User Interfaces, 2004 Available: http://www.isys.ucl.ac.be/bchi/members/bmi/pu blications.htm

[57] DRAFT "The Relationship of UIML 3.0 Spec. to Other Standards/Working Groups" July 18, 2003. Available: http://www.oasis-open.org/committees/ download.php/3419/

[58] G. D. Abowd, J. P.Bowen, A. J. Dix, M. D. Harrison, and R. Took. "User interface languages: A survey of existing methods." Technical Report PRG-TR-5-89, Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford, UK, October 1989

[59] Allan Meng Krebs, Ivan Marsic "Adaptive applications for ubiquitous collaboration" Rutgers — The State University of New Jersey Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 1, January 05 - 08, 2004, Big Island, Hawaii

[60] Jin Jing, Karen Huff, Himanshu Sinha, Ben Hurwitz, Bill Robinson "Workflow and application adaptations in mobile environments" GTE Laboratories Incorporated, Second IEEE Workshop on Mobile Computer Systems and Applications, February 25 - 26, 1999, New Orleans, Louisiana

[61] User Interface Markup Language (UIML) 3.0 Specification DRAFT, Document version 08 February 2002. http://www.uiml.org/specs/docs/uiml30-revised-02-12-02.pdf

[62] Abrams, M: UIML Tutorial. Available at http://www.harmonia.com, Accessed in January 2004

[63] Brad Myers, Scott E. Hudson, Randy Pausch, "Past, present, and future of user interface software tools" ACM Transactions on Computer-Human Interaction (TOCHI), Volume 7 , Issue 1 (March 2000), Special issue on human-computer interaction in the new millennium, Part 1, Pages: 3 - 28

[64] Tim clerkx, kris luyten, Karin coninx, "Generating context-sensitive multiple device interfaces from design" In Robert K. Jacob, and Quentin Limbourg, editors, Pre-Proceedings of the Fourth International Conference on Computer-Aided Design of User Interfaces CADUI'2004 jointly with the annual International Conference on Intelligent User Interfaces, January 13-16 2004, Funchal, Isle of Madeira, Portugal, pages 288-301, 2004. Available:

http://research.edm.luc.ac.be/~kris/research/publications/cadui2004/clerckxluyten
-cadui2004.pdf

[65] Nathalie Souchon, Quentin Limbourg, and Jean Vanderdonckt "Task Modelling in Multiple Contexts of Use" DSV-IS 2002, LNCS 2545, pp. 59-73, 2002.

[66] F.Paternò, C.Mancini, "Model-Based Design of Interactive Applications" ACM Intelligence, Winter 2000, pp.27-37.

[67] Wolfgang Mueller, Robbie Schaefer, Steffen Bluel "Interactive multimodal user interfaces for mobile devices" Paderborn University, January 05 - 08, 2004 Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) IEEE 2004 p90286

[68] Olsen, D.R., Jefferies, S., Nielsen, T., Moyes, W., and Fredrickson, P., "Cross-modal interaction using XWeb", Proceedings of UIST '00, San Deigo, CA, USA, November, 2000, pp.191-200

[69] Satyanarayanan, M., "Accessing information on demand at any location, Mobile Information Access" IEEE Personal Communications 3 (1), February, 1996. pp. 26-33

[70] Crease, M., Gray, P. and Brewster, S. (2000) A toolkit Mechanism and Context Independent Widgets. Proceedings of ISCE Workshop on Design, Specification and Verification of Interactive system DSVIS, June 5-6, 2000, Limerick, Ireland. Springer Verlag.

[71] Patricia Dockhorn Costa. Masters Thesis on "Towards a Services Platform for Context-Aware Applications" http://arch.cs.utwente.nl/assignments/thesis/ARCH-2003-04.pdf, August 2003

[72] AAIML: "Universal Remote Console Prototyping of an Emerging XML Based Alternate User Interface Access Standard." By Gottfried Zimmermann, Gregg Vanderheiden, and Al Gilman. Presented at the The Eleventh International World Wide Web Conference, 7-11 May 2002, Honolulu, Hawaii.

[73] Ronald A. Merrick, "AUIML: An XML Vocabulary for Describing User Interfaces. Device Independent User Interfaces in XML" Available at http://xml.coverpages.org/userInterfaceXML.html#auiml. May 09, 2001

[74] AUIT: John Grundy, Biao Yang "An environment for developing adaptive, multi-device user interfaces" Fourth Australasian User Interface Conference (AUIC 2003), Adelaide, Australia. Conferences in Research and Practice in Information Technology, Vol 18.

[75] "Object Modeling for User Centered Development and User Interface Design: The Wisdom Approach" By Duarte Nuno Jardim Nunes (Departamento de Matemática, Universidade da Madeira). April 2001

[76] "XIML: A Common Representation for Interaction Data.". By Angel Puerta and Jacob Eisenstein. Presented at the Sixth Intelligent User Interfaces Conference (IUI 2002), January 13-16, 2002, San Francisco, California, USA

[77] XIML Forum: http://www.ximl.org/ Accessed in March 2004

[78] "XML User Interface Language (XUL). Home page for Mozilla's XUL documentation" Available at http://luxor-xul.sourceforge.net/. Accessed in March 2004

[79] Abrams, M., Phanouriou, C. Batongbacal, A.L., Williams, S., and Shuster, J.E. (1998) UIML: An Appliance-Independent XML User Interface Language, In Proceedings of the 8th World Wide Web Conference, Toronto, Canada

[80] "Microsoft Extensible Application Markup Language (XAML)."Available: http://xmlcoverpages.org/ms-xaml.html. Accessed in April 2004

[81] Luyten, K., Van Laerhoven, T., Coninx, K., Van Reeth, F., 2003. "Runtime transformations for modal independent user interface migration. Interacting with Computers"

[82] W3C Device Independence Working Group (DIWG) Available: http://www.w3.org/2001/di/. Accessed in April 2004

[83] "XForms – The next generation of web forms" W3C XForms Working Group. Available: http://www.w3.org/MarkUp/Forms/ Accessed in March 2004

[84] Costin Pribeanu, Quentin Limbourg, Jean Vanderdonckt, "Task Modelling for Context-Sensitive User Interfaces" Available at http://www.dcs.gla.ac.uk/~johnson/papers/dsvis_2001/pribeaunu/ DSVIS, 13 June 2001

[85] Bomsdorf, B., Szwillus, G.: From Task to Dialogue: Task-Based User Interface Design. SIGCHI Bulletin 30, 4 1998 40–42. Accessible at http://www.acm.org/sigchi/bulletin/ 1998.4/szwillus.html

[86] Paternò, F.: Model Based Design and Evaluation of Interactive Applications. Springer Verlag, Berlin 1999

[87] Szekely, P.: Retrospective and Challenges for Model-Based Interface Development. In: Vanderdonckt, J.: Proc. of 2nd Int. Workshop on Computer-Aided Design of User Interfaces CADUI'96 (Namur, June 5-7, 1996). Presses Universitaires de Namur, Namur 1996 xxi–xliv. Accessible at http://www.isi.edu/isd/Mastermind/Internal/Files/SVIS96/ aper.ps.Z

[88] Tim Clerkx, Karin Conix "Integrating Task Models in Automatic User Interface Generation" Technical report. TR-LUC-EDM-0302, November 2003, EDM/LUC, Diepenbeek, Belgium.

[89] D. Navarre, P. Palanque, F. Paterno, C. Santoro, R. Bastide "A Tool Suite for Integrating Task and System Models through Scenarios" 8th Eurographics Workshop on Design, Specification and Verification of Interactive Systems, DSV-IS'2001, June 13-15, 2001. Lecture Notes in Computer Science, no. To appear. Glasgow, Scotland, Springer, 2001

[90] USIXML web page: http://www.usixml.org web site opened on May 15, 2004.

[91] Quentin Limbourg, Jean Vanderdonckt, Benjamin Michotte, Laurent Bouillon, Murielle Florins, and Daniela Trevisan "USIXML: A User Interface Description

Language for Context-Sensitive User Interfaces" Workshop organised at Advanced Visual Interfaces 2004, 25 May 2004.

[92] Souchon, N., Vanderdonckt, J., A Review of XML-Compliant User Interface Description Languages, Proc. of 10th Int. Conf. on Design, Specification, and Verification of Interactive Systems DSV-IS'2003 (Madeira, 4-6 June 2003), Jorge, J., Nunes, N.J., Falcao e Cunha, J. (Eds.), Lecture Notes in Computer Science, Vol. 2844, Springer-Verlag, Berlin, 2003, pp. 377-391.

[93] Silvia Berti, Francesco Correani, Fabio Paternò, Carmen Santoro. "The TERESA XML Language for the Description of Interactive Systems at Multiple Abstraction Levels" Workshop organized at Advanced Visual Interfaces AVI 2004

[94] Marc Abrams, Jim Helms. "Retrospective on UI Description Languages, Based on 7 Years Experience with the User Interface Markup Language (UIML)" Workshop organized at Advanced Visual Interfaces AVI 2004

[95] Mir Farooq Ali, Manuel A. Perez Quinones "Using Task models to generate Multi-Platform User Interfaces while ensuring Usability" CHI '02 extended abstracts on Human factors in computing systems, Pages 670-671

[96] Keith Cheverst, Keith Mitchell, Nigel Davies "The role of adaptive hypermedia in a context-aware tourist guide" Communications of the ACM, Volume 45, issue 5 May 2002, Special Issue; The adaptive web, pp. 47-51

[97] Peter Brusilovsky, Mark T. Maybury "From adaptive hypermedia to the adaptive web" Communications of the ACM, Volume 45, issue 5, May 2002, Special Issue: The adaptive web, pp.31-33

[98] Fabio Paternò "Model-Based Design and Evaluation of Interactive Applications" Springer-Verlag London Limited 2000. ISBN: 1-85233-155-0

[99] Paulo Pinheiro da Silva, Norman W. Paton: "UMLi: The Unified Modeling Language for Interactive Applications," p. 117-132, 2-6 October, 2000, York, UK

[100] "UMLi: Unified Modeling Language for interactive applications" home page available: http://www.cs.man.ac.uk/img/umli/. Accessed in March 2004

[101] Jan Van den Bergh, Kris Luyten, Karin Coninx "A Run-time System for Context-Aware Multi-Device User Interfaces" HCI International 2003, Volume 2, pp. 308-312, Crete, Greece, 2003.

[102] Jan Van den Bergh and Karin Coninx, "Contextual ConcurTaskTrees: Integrating dynamic contexts in task based design" CoMoRea 2004, workshop at Percom 2004, Orlando, CA, USA. Second IEEE Conference on Pervasive Computing and Communications Workshops , pp. 13-17, Orlando, USA, 2004

[103] D. Diaper, Neville A. Stanton "The Handbook of Task Analysis for Human-Computer Interaction" Lawrence Erlbaum Assoc (September, 2003) ISBN: 0805844333

[104] GIML/GITK Architecture Available: http://gitk.sourceforge.net/architecture.html Accessed in May 2004

[105]Giulio Mori, Fabio Paterno and Carmen Santoro "CTTE: Support for Developing and Analyzing Task Models for Interactive System Design" IEEE Transactions on software engineering, Vol 28, No. 9, September 2002.

APPENDIX A – USER'S CURRENT CONTEXT SPECIFICATION LANGUAGE

Schema definition for user's current context specification language

```xml
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<!--    Author:          Created by Reena G. Hanumansetty
        About:           Formal Description of User's current Context
        History:
                         Created: June 2nd, 2004
-->
<!--    User's current context Description -->
<xs:element name="UsersCurrentContext" type="uccType"/>

<!-- ********************************************* -->
<!-- Simple Type Definitions-->

<xs:complexType name="uccType">
        <xs:sequence>
                <xs:element name="user" type="contextType"/>
                <xs:element name="platform" type="contextType"/>
                <xs:element name="environment" type="contextType"/>
        </xs:sequence>
        <xs:attribute name="ctxEntityID" type="xs:string"
        use="required"/>
        <!-- It represents unique user or object associated with the
        current context description -->
        <!-- ********* Uniqueness has to be incorporated into the
        schema********** -->
        <xs:attribute name="timeStamp" type="xs:time" use="required"/>
</xs:complexType>

<xs:simpleType name="lvlType">
        <xs:restriction base="xs:string">
        <xs:enumeration value="content"/>
        <xs:enumeration value="task"/>
        <xs:enumeration value="abstractUI"/>
        <xs:enumeration value="concreteUI"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="natureType">
        <xs:restriction base="xs:string">
        <xs:enumeration value="dynamic"/>
        <xs:enumeration value="static"/>
        </xs:restriction>
</xs:simpleType>
```

```xml
<!--      Complex Type definitions -->

<xs:complexType name="paramType">
        <xs:sequence>
                <xs:element name="value" type="xs:string"/>
                <xs:element name="effectedUILevels" type="levelType"
                maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string"/>
        <xs:attribute name="ctxNature" type="natureType"
        use="required"/>
</xs:complexType>


<xs:complexType name="levelType">
        <xs:sequence>
                <xs:element name="level" type="lvlType"
                maxOccurs="4"/>
        </xs:sequence>
</xs:complexType>
<xs:complexType name="ctxType">
        <xs:sequence>
        <xs:element name="ctxParameter" type="paramType"
                minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string" use="required"/>
</xs:complexType>


<xs:complexType name="contextType">
        <xs:sequence>
                <xs:element name="context" type="ctxType"
                maxOccurs="unbounded" minOccurs="0"/>
        </xs:sequence>
</xs:complexType>

<!-- ********************************************* -->
</xs:schema>
```

Figure A: Schema definition for UCC (User's current context)

APPENDIX B – CONCUR TASK TREE MODEL

Schema Definition for Concur Task Tree model

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<!-- Main tree description -->
<xs:element name="TaskModel">
        <xs:complexType>
                <xs:element name="Task" type="Tasktype"/>
        </xs:complexType>
</xs:element>

<!-- Complex Type definitions -->
<xs:complexType name="TaskType">
        <xs:sequence>
                <xs:element name="Name" type="StringType"/>
                <xs:element name="Type" type="StringType"/>
                <xs:element name="Description" type="StringType"/>
                <xs:eleemnt name="Platform" type="StringType"
                maxOccurs="unbounded"/>
                <xs:element name="Precondition" type="StringType"
                minOccurs="0" maxOccurs="1"/>
                <xs:element name="TemporalOperator"
                type="temporalOpType" minOccurs="0" maxOccurs="1"/>
                <xs:element name="TimePerformance" type="timeType"/>
                <xs:element name="Parent" type="nodeNameType"
                minOccurs="0" maxOccurs="1"/>
                <xs:element name="SiblingLeft" type="nodeNameType"
                minOccurs="0" maxOccurs="1"/>
                <xs:element name="SiblingRight" type="nodeNameType"
                minOccurs="0" maxOccurs="1"/>
                <xs:element name="Object" type="objectType"
                maxOccurs="unbounded"/>
                <xs:element name="SubTask" type="subTaskType"
                maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="Identifier" type="StringType"
                      use="required"/>
        <xs:attribute name="Category" type="catType"
                      use="required"/>
        <xs:attribute name="Iterative" type="iterativeType"
                      use="required"/>
        <xs:attribute name="Optional" type="optionalType"
                      use="required"/>
        <xs:attribute name="PartOfCooperation" type="coopType"
                      use="required"/>
```

```xml
            <xs:attribute name="Frequency" type="freqType"
                          use="required"/>
</xs:complexType>

<xs:complexType name="temporalOpType">
        <xs:attribute name="name" type="temporalType" use="implied"/>
</xs:complexType>

<xs:complexType name="TimePerformance">
        <xs:sequence>
        <xs:element name="Max" type="StringType"/>
        <xs:element name="Min" type="StringType"/>
        <xs:element name="Average" type="StirngType"/>
        </xs:sequence>
</xs:complexType>

<xs:complexType name="nodeNameType">
        <xs:attribute name="name" type="StringType"/>
</xs:complexType>
<xs:complexType name="ObjectType">
        <xs:sequence>
                <xs:element name="Platform" minOccurs="0"
                maxOccurs="unbounded"/>
                <xs:element name="inputAction" type="inpactionType"/>
                <xs:element name="outputAction" type="outactionType"/>
        </xs:sequence>
        <xs:attribute name="name" type="StringType"/>
        <xs:attribute name="class" type="classType"/>
        <xs:attribute name="type" type="typeOfObject"/>
        <xs:attribute name="access_mode" type="accessType"/>
        <xs:attribute name="cardinality" type="cardinalType"/>
</xs:complexType>
<xs:complexType name="inpactionType">
        <xs:sequence>
                <xs:attribute name="Description" type="StringType"/>
                <xs:attribute name="From" type="StringType"/>
        </xs:sequence>
</xs:complexType>

<xs:complexType name="outactionType">
        <xs:sequence>
                <xs:attribute name="Description" type="StringType"/>
                <xs:attribute name="To" type="StringType"/>
        </xs:sequence>
</xs:complexType>

<xs:complexType name="SubTask">
        <xs:element name="Task" type="TaskType" minOccurs="0"
        maxOccurs="unbounded"/>
</xs:complexType>
```

```
<!-- Simple Type definitions -->
<xs:simpleType name="catType">
        <xs:restriction base="xs:string">
        <xs:enumeration value="abstraction"/>
        <xs:enumeration value="user"/>
        <xs:enumeration value="interaction"/>
        <xs:enumeration value="application"/>
        </xs:restriction>
</xs:simpleType>

<xs:simpleType name="interativeType">
        <xs:restriction base="xs:string">
        <xs:enumeration value="true"/>
        <xs:enumeration value="false"/>
        </xs:restriction>
</xs:simpleType>

<xs:simpleType name="optionalType">
        <xs:restriction base="xs:string">
        <xs:enumeration value="true"/>
        <xs:enumeration value="false"/>
        </xs:restriction>
</xs:simpleType>

<xs:simpleType name="coopType">
        <xs:restriction base="xs:string">
        <xs:enumeration value="true"/>
        <xs:enumeration value="false"/>
        </xs:restriction>
</xs:simpleType>

<xs:simpleType name="freqType">
        <xs:restriction base="xs:integer">
        <xs:minInclusive value="0"/>
        <xs:maxInclusive value="10"/>
        </xs:restriction>
</xs:simpleType>

<xs:simpleType name="temporalType">
        <xs:restriction base="xs:string">
        <xs:enumeration value="SequentialEnabling"/>
        <xs:enumeration value="Disabling"/>
        <xs:enumeration value="Choice"/>
        <xs:enumeration value="Interleaving"/>
        <xs:enumeration value="Synchronization"/>
        <xs:enumeration value="SuspendResume"/>
        <xs:enumeration value="SequentialEnablingInfo"/>
        </xs:restriction>
</xs:simpleType>
```

```xml
<xs:simpleType name="classType">
        <xs:restriction base="xs:string">
        <xs:enumeration value="Text"/>
        <xs:enumeration value="Numerical"/>
        <xs:enumeration value="Graphic"/>
        <xs:enumeration value="Image"/>
        <xs:enumeration value="Position"/>
        <xs:enumeration value="null"/>
        </xs:restriction>
</xs:simpleType>

<xs:simpleType name="typeOfObject">
        <xs:restriction base="xs:string">
        <xs:enumeration value="Perceivable"/>
        <xs:enumeration value="Application"/>
        <xs:enumeration value="null"/>
        </xs:restriction>
</xs:simpleType>

<xs:simpleType name="accessType">
        <xs:restriction base="xs:string">
        <xs:enumeration value="Access"/>
        <xs:enumeration value="Modification"/>
        <xs:enumeration value="null"/>
        </xs:restriction>
</xs:simpleType>

<xs:simpleType name="cardinalType">
        <xs:restriction base="xs:string">
        <xs:enumeration value="Low"/>
        <xs:enumeration value="Medium"/>
        <xs:enumeration value="High"/>
        <xs:enumeration value="null"/>
        </xs:restriction>
</xs:simpleType>

</xs:schema>
```

Figure B: Schema Definition for ConcurTaskTree Model

APPENDIX C – CONCUR CONTEXT TREE MODEL

Schema Definition for Concur Context Tree Model

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<!-- Simple Type definitions -->
<xs:simpleType name="typeOfCondition">
        <xs:restriction base="xs:string">
                <xs:enumeration value="and"/>
                <xs:enumeration value="or"/>
                <xs:enumeration value="equal"/>
                <xs:enumeration value="lt"/>
                <xs:enumeration value="gt"/>
        </xs:restriction>
</xs:simpleType>

<xs:simpleType name="catType">
        <xs:restriction base="xs:string">
        <xs:enumeration value="abstraction"/>
        <xs:enumeration value="user"/>
        <xs:enumeration value="interaction"/>
        <xs:enumeration value="application"/>
        <xs:enumeration value="choice"/>
        </xs:restriction>
</xs:simpleType>

<xs:simpleType name="interativeType">
        <xs:restriction base="xs:string">
        <xs:enumeration value="true"/>
        <xs:enumeration value="false"/>
        </xs:restriction>
</xs:simpleType>

<xs:simpleType name="optionalType">
        <xs:restriction base="xs:string">
        <xs:enumeration value="true"/>
        <xs:enumeration value="false"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="coopType">
        <xs:restriction base="xs:string">
        <xs:enumeration value="true"/>
        <xs:enumeration value="false"/>
        </xs:restriction>
</xs:simpleType>
```

```xml
<xs:simpleType name="freqType">
        <xs:restriction base="xs:integer">
        <xs:minInclusive value="0"/>
        <xs:maxInclusive value="10"/>
        </xs:restriction>
</xs:simpleType>

<xs:simpleType name="temporalType">
        <xs:restriction base="xs:string">
        <xs:enumeration value="SequentialEnabling"/>
        <xs:enumeration value="Disabling"/>
        <xs:enumeration value="Choice"/>
        <xs:enumeration value="Interleaving"/>
        <xs:enumeration value="Synchronization"/>
        <xs:enumeration value="SuspendResume"/>
        <xs:enumeration value="SequentialEnablingInfo"/>
        </xs:restriction>
</xs:simpleType>

<xs:simpleType name="classType">
        <xs:restriction base="xs:string">
        <xs:enumeration value="Text"/>
        <xs:enumeration value="Numerical"/>
        <xs:enumeration value="Graphic"/>
        <xs:enumeration value="Image"/>
        <xs:enumeration value="Position"/>
        <xs:enumeration value="null"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="typeOfObject">
        <xs:restriction base="xs:string">
        <xs:enumeration value="Perceivable"/>
        <xs:enumeration value="Application"/>
        <xs:enumeration value="null"/>
        </xs:restriction>
</xs:simpleType>

<xs:simpleType name="accessType">
        <xs:restriction base="xs:string">
        <xs:enumeration value="Access"/>
        <xs:enumeration value="Modification"/>
        <xs:enumeration value="null"/>
        </xs:restriction>
</xs:simpleType>

<xs:simpleType name="cardinalType">
        <xs:restriction base="xs:string">
        <xs:enumeration value="Low"/>
        <xs:enumeration value="Medium"/>
```

```xml
            <xs:enumeration value="High"/>
            <xs:enumeration value="null"/>
            </xs:restriction>
</xs:simpleType>

<xs:simpleType name="StringType">
            <xs:restriction base="xs:string"/>
</xs:simpleType>

<!-- Complex Type definitions -->
<xs:complexType name="conditionType">
            <xs:choice>
                    <xs:group ref="contextValuePair"/>
                    <xs:element name="condition" type="conditionType"/>
            </xs:choice>
            <xs:attribute name="condtype" type="typeOfCondition"/>
</xs:complexType>
<xs:complexType name="choiceTask">
            <xs:sequence>
                    <xs:element name="condition" type="conditionType"/>
                    <xs:element name="SubTask" type="TaskType"
                    minOccurs="2" maxOccurs="2"/>
            </xs:sequence>
            <xs:attribute name="Identifier" type="StringType"
            use="required"/>
            <xs:attribute name="Category" type="StringType"
            fixed="choiceNode"/>
</xs:complexType>

<xs:group name="contextValuePair">
            <xs:sequence>
            <xs:element name="context" type="contextType"/>
            <xs:element name="value" type="StringType"/>
            </xs:sequence>
</xs:group>

<xs:complexType name="contextType">
            <xs:attribute name="name" type="StringType" use="required"/>
            <xs:attribute name="attname" type="StringType"
            use="required"/>
</xs:complexType>

<xs:complexType name="simpleTaskType">
            <xs:sequence>
                    <xs:element name="Name" type="StringType"/>
                    <xs:element name="Type" type="StringType"/>
```

```xml
                    <xs:element name="Description" type="StringType"/>
                    <xs:element name="Platform" type="StringType"
                                maxOccurs="unbounded"/>
                    <xs:element name="Precondition" type="StringType"
                                minOccurs="0" maxOccurs="1"/>


                    <xs:element name="TemporalOperator"
                                type="temporalOpType"
                                minOccurs="0" maxOccurs="1"/>
                    <xs:element name="TimePerformance"
                                type="StringType"/>
                    <xs:element name="Parent" type="nodeNameType"
                                minOccurs="0" maxOccurs="1"/>
                    <xs:element name="SiblingLeft" type="nodeNameType"
                                minOccurs="0" maxOccurs="1"/>
                    <xs:element name="SiblingRight" type="nodeNameType"
                                minOccurs="0" maxOccurs="1"/>
                    <xs:element name="Object" type="objectType"
                                maxOccurs="unbounded"/>
                    <xs:element name="SubTask" type="TaskType"
                                minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="Identifier" type="StringType"
                      use="required"/>
        <xs:attribute name="Category" type="catType"
                      use="required"/>
        <xs:attribute name="Iterative" type="StringType"
                      use="required"/>
        <xs:attribute name="Optional" type="optionalType"
                      use="required"/>
        <xs:attribute name="PartOfCooperation" type="coopType"
                      use="required"/>
        <xs:attribute name="Frequency" type="freqType"
                      use="required"/>
        <xs:attribute name="Active" type="xs:boolean"
                      use="required"/>
</xs:complexType>


<xs:complexType name="TaskType">
        <xs:choice>
        <xs:element name="Task" type = "simpleTaskType"/>
        <xs:element name="ChoiceNode" type="choiceTask"/>
        </xs:choice>
</xs:complexType>

<xs:complexType name="temporalOpType">
        <xs:attribute name="name" type="temporalType" />
</xs:complexType>
```

87

```xml
<xs:complexType name="TimePerformance">
        <xs:sequence>
        <xs:element name="Max" type="StringType"/>
        <xs:element name="Min" type="StringType"/>
        <xs:element name="Average" type="StringType"/>
        </xs:sequence>
</xs:complexType>

<xs:complexType name="nodeNameType">
        <xs:attribute name="name" type="StringType"/>
</xs:complexType>

<xs:complexType name="objectType">
        <xs:sequence>
                <xs:element name="Platform" minOccurs="0"
                            maxOccurs="unbounded"/>
                <xs:element name="inputAction"
                            type="inpactionType"/>
                <xs:element name="outputAction"
                            type="outactionType"/>
        </xs:sequence>
        <xs:attribute name="name" type="StringType"/>
        <xs:attribute name="class" type="classType"/>
        <xs:attribute name="type" type="typeOfObject"/>
        <xs:attribute name="access_mode" type="accessType"/>
        <xs:attribute name="cardinality" type="cardinalType"/>
</xs:complexType>

<xs:complexType name="inpactionType">
        <xs:attribute name="Description" type="StringType"/>
        <xs:attribute name="From" type="StringType"/>
</xs:complexType>

<xs:complexType name="outactionType">
        <xs:attribute name="Description" type="StringType"/>
        <xs:attribute name="To" type="StringType"/>
</xs:complexType>

<!-- Main tree description -->
<xs:element name="TaskModel" type="TaskType"/>

</xs:schema>
```

Figure C: Schema Definition for Concur Context Tree Model

APPENDIX D – RULE SPECIFICATION

Schema definition for Rule Specification

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<!-- targetNamespace="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.w3schools.com"-->

<!-- Simple Type definitions -->

<xs:simpleType name="rtType">
        <xs:restriction base="xs:string">
                <xs:enumeration value="static"/>
                <xs:enumeration value="dyamic"/>
        </xs:restriction>
</xs:simpleType>

<xs:simpleType name="plasticType">
        <xs:restriction base="xs:string">
                <xs:enumeration value="user"/>
                <xs:enumeration value="platform"/>
                <xs:enumeration value="environment"/>
        </xs:restriction>
</xs:simpleType>

<xs:simpleType name="typeOfCondition">
        <xs:restriction base="xs:string">
                <xs:enumeration value="and"/>
                <xs:enumeration value="or"/>
                <xs:enumeration value="equal"/>
                <xs:enumeration value="notEqual"/>
                <xs:enumeration value="lt"/>
                <xs:enumeration value="gt"/>
        </xs:restriction>
</xs:simpleType>

<xs:simpleType name="StringType">
        <xs:restriction base="xs:string"/>
</xs:simpleType>

<!-- complex Type descriptions -->
<xs:complexType name="cParamType">
        <xs:sequence>
                <xs:element name="value" type="StringType"/>
        </xs:sequence>
        <xs:attribute name="name" type="StringType" use="required"/>
</xs:complexType>
```

```
<xs:complexType name="contextType">
        <xs:choice>
                <xs:element name="ctxParameter"  type="cParamType"
                            minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="value" type="StringType"/>
        </xs:choice>
        <xs:attribute name="type" type="plasticType"
                      use="required"/>
        <xs:attribute name="ctxName" type="StringType"
                      use="required"/>
</xs:complexType>

<xs:complexType name="conditionType">
        <xs:choice>
                <xs:element name="context" type="contextType"/>
                <xs:element name="condition" type="conditionType"
                            maxOccurs="unbounded"/>
        </xs:choice>
        <xs:attribute name="type" type="typeOfCondition"
                      use="required"/>
</xs:complexType>

<xs:complexType name="taskType">
        <xs:attribute name="id" type="StringType"/>
</xs:complexType>

<xs:complexType name="taskListType">
        <xs:sequence>
                <xs:element name="task" type="taskType"
                            minOccurs="0" maxOccurs="unbounded"/>

        </xs:sequence>
        <xs:attribute name="all" type="xs:boolean" default="false"/>
</xs:complexType>

<xs:complexType name="idType">
        <xs:attribute name="type" type="StringType" use="required"/>
        <xs:attribute name="value" type="StringType"
                      use="required"/>
</xs:complexType>

<xs:complexType name="idListType">
        <xs:sequence>
                <xs:element name="id" type="idType"
                            maxOccurs="unbounded"/>
        </xs:sequence>
</xs:complexType>
```

```
<xs:complexType name="actionType">
        <xs:sequence>
                <xs:element name="idList" type="idListType"/>
                <xs:element name="change" type="chType"/>
        </xs:sequence>
</xs:complexType>

<xs:complexType name="chType">
        <xs:sequence>
                <xs:element name="param" type="StringType"
                                maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="type" type="StirngType" use="optional"/>
</xs:complexType>

<xs:complexType name="taskActionType">
        <xs:sequence>
        <xs:element name="taskList" type="taskListType"/>
        <xs:element name="change" type="chType"/>
        </xs:sequence>
        <xs:attribute name="level" type="levelType" use="required"/>
        </xs:sequence>
        <xs:attribute name="level" type="levelType" use="required"/>
</xs:complexType>



<xs:complexType name="actionsType">
        <xs:choice>
                <xs:element name="TaskLevelAction"
                type="taskActionType" maxOccurs="unbounded"/>
                <xs:element name="contentLevelAction"
                type="actionType" maxOccurs="unbounded"/>
                <xs:element name="abstractLevelAction"
                type="actionType" maxOccurs="unbounded"/>
                <xs:element name="concreteLevelAction"
                type="actionType" maxOccurs="unbounded"/>
        </xs:choice>
</xs:complexType>

<xs:complexType name="ruleType">
        <xs:sequence>
                <xs:element name="condition" type="conditionType"/>
                <xs:element name="actions" type="actionsType"/>
        </xs:sequence>
        <xs:attribute name="type" type="rtType"/>
</xs:complexType>
```

91

```
<xs:complexType name="rulesType">
        <xs:sequence>
                <xs:element name="rule" type="ruleType"
                            maxOccurs="unbounded"/>
        </xs:sequence>
</xs:complexType>

<!-- root description -->
<xs:element name="rules" type="rulesType"/>

</xs:schema>
```

Figure D: Schema Definition for Rule Specification

APPENDIX E – EXAMPLE VXML FILE

For the case Study of Chapter 4, initial interaction menu of the application is shown in the vxml file:

```xml
<?xml version="1.0" encoding="iso-8859-1" ?>
- <vxml version="2.0">
    <property name="confidencelevel" value="0.5" />
    <property name="sensitivity" value="0.5" />
    <property name="completetimeout" value="1" />
    <property name="incompletetimeout" value="3" />
    <property name="bargein" value="false" />
  - <form>
    - <block>
      - <prompt count="1">
          <prosody pitch="1" rate="1" volume="100">. Remember that you can always use ciao to exit from
            application.</prosody>
        </prompt>
        <prompt count="2">.</prompt>
        <goto next="#Ordering_1_0" />
      </block>
    </form>
  - <form id="Ordering_1_0">
    - <block>
        At first,
        <goto next="#Select_Transportation_service1" />
      </block>
    </form>
  - <menu id="Select_Transportation_service1">
    - <prompt>
        <prosody pitch="1" rate="1" volume="100">To Select Transportation service, say Transportation;</prosody>
      </prompt>
      <choice next="#Transportation_choice">Transportation</choice>
      <prompt>Then,</prompt>
    - <prompt>
```

```
- <prompt>
    <prosody pitch="1" rate="1" volume="100">To Select Home Repairs, say Home Repairs;</prosody>
  </prompt>
  <choice next="#Home_Repairs_choice">Home Repairs</choice>
  <prompt>Then,</prompt>
- <prompt>
    <prosody pitch="1" rate="1" volume="100">To Select Access to health information, say health
      info;</prosody>
  </prompt>
  <choice next="#health_info_choice">health info</choice>
  <prompt>Then,</prompt>
- <prompt>
    <prosody pitch="1" rate="1" volume="100">To Order Meals, say Meals;</prosody>
  </prompt>
  <choice next="#Meals_choice">Meals</choice>
  <prompt>Then,</prompt>
- <prompt>
    <prosody pitch="1" rate="1" volume="100">For Insurance Counceling, say Counceling;</prosody>
  </prompt>
  <choice next="#Counceling_choice">Counceling</choice>
  <prompt>Lastly,</prompt>
- <prompt>
    <prosody pitch="1" rate="1" volume="100">For advice on Legal Issues and services, say Legal
      Issues;</prosody>
  </prompt>
  <choice next="#Legal_Issues_choice">Legal Issues</choice>
  <choice next="#ciao">ciao</choice>
</menu>
- <form id="Transportation_choice">
  - <block>
    - <prompt>
        <prosody pitch="1" rate="1" volume="100">Ok, loading Transportation.</prosody>
      </prompt>
      <goto next="presentation_2.vxml" />
```

```
        </block>
      </form>
    - <form id="Home_Repairs_choice">
      - <block>
        - <prompt>
            <prosody pitch="1" rate="1" volume="100">Ok, loading Home Repairs.</prosody>
          </prompt>
          <goto next="presentation_4.vxml" />
        </block>
      </form>
    - <form id="health_info_choice">
      - <block>
        - <prompt>
            <prosody pitch="1" rate="1" volume="100">Ok, loading health info.</prosody>
          </prompt>
          <goto next="presentation_6.vxml" />
        </block>
      </form>
    - <form id="Meals_choice">
      - <block>
        - <prompt>
            <prosody pitch="1" rate="1" volume="100">Ok, loading Meals.</prosody>
          </prompt>
          <goto next="presentation_8.vxml" />
        </block>
      </form>
    - <form id="Counceling_choice">
      - <block>
        - <prompt>
            <prosody pitch="1" rate="1" volume="100">Ok, loading Counceling.</prosody>
          </prompt>
```

```
            <goto next="presentation_14.vxml" />
          </block>
        </form>
      - <form id="Legal_Issues_choice">
        - <block>
          - <prompt>
              <prosody pitch="1" rate="1" volume="100">Ok, loading Legal Issues.</prosody>
            </prompt>
            <goto next="presentation_15.vxml" />
          </block>
        </form>
      - <form id="ciao">
        - <block>
          - <prompt>
              <prosody pitch="1" rate="1" volume="100">Thank you for visting Untitled voice application. Good Bye!
                </prosody>
            </prompt>
            <disconnect />
          </block>
        </form>
    </vxml>
```

Figure E: Example Voice XML

APPENDIX F – EXAMPLE RULES FILE

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- action.xml -->
<rules xmlns="http://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="actionSpecification2.xsd">
<!-- RULE1 -->
<rule type="static">
    <condition type="and">
        <condition type="equal">
            <context type="platform" ctxName="platformName">
                <value>Desktop</value>
            </context>
        </condition>
                <condition type="equals">
                <context type="user" ctxName="preferences">
                    <ctxParameter name="Navigator Choice">
                            <value>Graphic Link</value>
                    </ctxParameter>
                </context>
            </condition>
        </condition>
        <actions>
            <concreteLevelAction>
                <idlist>
                    <id type="task" value="Select_Transportation_service"/>
                    <id type="task" value="Select_Home_Reairs"/>
                    <id type="task" value="Select_Access_to_health_info"/>
                    <id type="task" value="Select_Order_Service"/>
                    <id type="task" value="Select_Counseling_service"/>
                    <id type="task" value="Select_Legal_Issues_and_services"/>
                </idlist>
                <change>
                    <param>view_navigation_label</param>
                    <param>graphic link</param>
                </change>
            </concreteLevelAction>
        </actions>
    </rule>
<!-- RULE 2-->
<rule type="dynamic">
    <condition type="equal">
        <context type="platform" ctxName="platformName">
            <value>PDA</value>
        </context>
    </condition>
    <actions>
```

```
    <TaskLevelAction>
      <tasklist all="true"/>
      <change type="changeAttribute">
          <param>platform</param>
          <param>PDA</param>
      </change>
    </TaskLevelAction>
   </actions>
</rule>

<!-- RULE3 -->
<rule type="dynamic">
    <condition type="or">
        <condition type="gt">
            <context type="user" ctxName="taskFrequencyIncrease">
                <ctxParameter name="Transportation">
                    <value>5</value>
                </ctxParameter>
            </context>
        </condition>
        <condition type="gt">
            <context type="user" ctxName="taskFrequencyIncrease">
                <ctxParameter name="Home Repairs">
                    <value>5</value>
                </ctxParameter>
            </context>
        </condition>
        <condition type="gt">
            <context type="user" ctxName="taskFrequencyIncrease">
                <ctxParameter name="Access Health Information">
                <ctxParameter name="Access Health Information">
                    <value>5</value>
                </ctxParameter>
            </context>
        </condition>
        <condition type="gt">
            <context type="user" ctxName="taskFrequencyIncrease">
                    <ctxParameter name="Order Meals">
                    <value>5</value>
                    </ctxParameter>
            </context>
        </condition>
        <condition type="gt">
            <context type="user" ctxName="taskFrequencyIncrease">
                    <ctxParameter name="Insurance Counseling">
                    <value>5</value>
                    </ctxParameter>
            </context>
        </condition>
```

```xml
              <condition type="gt">
                    <context type="user" ctxName="taskFrequencyIncrease">
                            <ctxParameter name="Legal Issues and Services">
                            <value>5</value>
                            </ctxParameter>
                    </context>
              </condition>
         </condition>
       <actions>
              <TaskLevelAction>
                 <taskList>
                        <task id="Transport Service"/>
                 </taskList>
                 <change type="restructureTree">
                        <param>sortSiblings</param>
                        <param>frequency</param>
                 </change>
              </TaskLevelAction>
       </actions>
</rule>
       <!-- RULE4 -->
<rule type="dynamic">
       <condition type="gt">
              <context type="user" ctxName="Behavior">
                        <ctxParameter name="MOuse Movement">
                        <value>300units</value>
                        </ctxParameter>
              </context>
       </condition>
       <actions>
              <TaskLevelAction>
                  <taskList>
                        <task id="currentTask"/>
                  </taskList>
                  <change type="restructureTree">
                        <param>AddTask</param>
                        <param>alertExtensiveMouseMotion</param>
                  </change>
              </TaskLevelAction>
       </actions>
</rule>
       <!-- RULE 5 -->
<rule type="dynamic">
       <condition type="and">
          <condition type="notEqual">
              <context type="environment" ctxName="Location">
                        <ctxParameter name="xco-ord">
                        <value>null</value>
                        </ctxParameter>
              </context>
```

```xml
                <context type="environment" ctxName="Location">
                        <ctxParameter name="yco-ord">
                        <value>null</value>
                        </ctxParameter>
                </context>
            </condition>
        </condition>
        <actions>
                <TaskLevelAction>
                    <taskList>
                            <task id="Provide request1"/>
                            <task id="Provide request2"/>
                            <task id="Enter Zip Code"/>
                            <task id="Provide request3"/>
                    </taskList>
                    <change type="changeAttribute">
                            <param>Active</param>
                            <param>false</param>
                    </change>
                </TaskLevelAction>
                <TaskLevelAction>
                    <taskList>
                            <task id="currentTask"/>
                    </taskList>
                    <change type="restructureTree">
                            <param>AddTask</param>
                            <param>confirmOrChangeLocation</param>
                    </change>
                </TaskLevelAction>
        </actions>
    </rule>
</rules>
```

Figure F: Rule fie for the scenario of chapter 4

VITA

Reena Gowri Hanumansetty, daughter of Mallikarjuna Rao Hanumansetty and Jyothi Hanumansetty, graduated from Gandhi Institute of Technology And Management, Visakhapatnam, India with a Bachelor of Technology degree in Computer Science and Engineering in May 2001. After working for 8 months in Gayatri Vidya Parishad College of Engineering, Visakhapatnam, as an Instructor, she came to the United States on July 30, 2002 to study at Virginia Tech. She is an invited member of Phi Kappa Phi Honor society. This thesis completes her Master's degree in Computer Science and Applications at Virginia Tech. She will begin a position as a Software Engineer with Advanced Simulation Technology Inc, Herndon, VA starting July 27, 2004.