

**AN EFFICIENT MODEL DRIVEN ARCHITECTURE  
APPROACH FOR CONTENT REPURPOSING**

by

K.Sivan Arul Selvan

Reg. No. 71203405016

of

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE - 641006.

A PROJECT REPORT

Submitted to the

FACULTY OF INFORMATION AND COMMUNICATION ENGINEERING

*In partial fulfillment of the requirements*

*for the award of the degree*

*of*

**MASTER OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

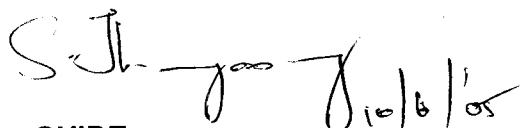
**June, 2005**



P-1543

## BONAFIDE CERTIFICATE

Certified that this project report titled **AN EFFICIENT MODEL DRIVEN ARCHITECTURE APPROACH FOR CONTENT REPURPOSING** is the bonafide work of **Mr.K.Sivan Arul Selvan** who carried out the research under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

  
GUIDE 10/6/05

  
HEAD OF THE DEPARTMENT 10/6/05

The candidate with University Register No. 71203405016 was examined by us in Project Viva-Voce examination held on 25/06/05

  
INTERNAL EXAMINER

  
EXTERNAL EXAMINER 25/6/05

## ABSTRACT

Content repurposing adjusts existing marked up content to enable its reuse for various device profiles and usage scenarios. Another way of defining is modifying the content or outlook or both according to user. Advantages of repurpose are reduction of effort, cost and time.

We know that a model is a coherent set of formal elements describing something (for example systems such as bank, phone, or train) built for some purpose that is amenable to a particular form of analysis, such as communication of ideas between people and machines, completeness checking, test case generation & verification, viability in terms of indicators such as cost estimation, standards, transformation into an implementation etc. A model need not be complete representation of entire system; it may represent one of the functional units of the system. Often, a graphical model excludes code; UML is such and best one.

The proposed solution is the multimedia metamodel, which introduces concepts and mechanisms from the multimedia domain. Content repurposing is defined by these elements content, containing markup, but no formatting information. A transformation engine that takes content and changes its format, info based on external rules that can be modified. The transformation engine does not alter the content but can add content based on external rules. An imaging engine that presents the formatted info based on external application-neutral style sheets.

The transformation of content can occur at several different places (on server / client side). I.e. the server can control the output of the content and the client can control the uptake of content.

This model driven approach is inspired by OMG's (Object Management Group) MDA (Model Driven Architecture) which facilitates system specification and interoperability using hierarchically organized formal models.

This report focuses about developing a data dictionary for ATM system development to illustrating the advantages of content repurposing and model driven approach with the help of models using UML and implementing it using XML.

## கருத்துச் சுருக்கம்

தகவல் மறுநோக்கம் (Content Repurposing) என்பது ஏற்கனவே உள்ள குறியீடு செய்யப்பட்ட (Markup) தகவல்களைப் புதிய நோக்கத்திற்காகவும், வேறுபட்ட கருவிகளுக்கு ஏற்பவும் மற்றும் உபயோகிப்பாளர்களுக்குத் தகுந்தவாறு மாற்றியமைத்தலாகும். மற்றொரு வகையில் தகவல் மறுநோக்கமானது உபயோகிப்பாளருக்கு தக்கவாறு தகவலையோ, தோற்றத்தையோ அல்லது இரண்டையுமோ மாற்றி அமைத்தலாகும். தகவல் மறுநோக்கமானது கீழ்க்கண்டவற்றின் தேவையை குறைத்து நமக்கு நன்மை பயப்பதாக உள்ளது. அவை முறையே உழைப்பு (Effort), நேரம் மற்றும் பணம்.

மாதிரியின் (Model) உபயோகம் நம் அனைவரும் அறிந்ததே. ஏதோ ஒன்றை (உதாரணமாக : வங்கியின் இயக்கம், கோள்களின் இயக்கம், புகைவண்டியின் செயல்பாடு) விளக்கத் தேவையான முக்கிய காரண-காரியம், தொடர்பு, செயல்பாடு போன்றவற்றை உள்ளடக்கியது. உதாரணத்திற்கு மனிதர்களுக்கும் - கருவிகளுக்கும் இடையேயான தகவல் பரிமாற்றம், சோதனைக்கான மாதிரியை உற்பத்தி செய்தல் (test case generation), பண மதிப்பீடு செய்தல் (Cost estimation), நியமங்களை (Standards) அமுலுக்கு கொண்டு வருதல் போன்றவற்றிற்கு பேருதவியாய் இருக்கும். ஒரு மாதிரியானது கண்டிப்பாக ஒரு முழு இயக்கத்தையும் விளக்குவதாக இருக்க வேண்டும் என்பது அவசியம் இல்லை. அது ஒரு இயக்கத்தின் ஒரு தனிப் பகுதியை விளக்குவதாக வேண்டுமானாலும் இருக்கலாம். வரைபட மாதிரிகள், கணிப்பொறி மொழிகளின் தேவையை (மாதிரியை குறிக்க) நீக்குகின்றன. ஒருங்கிணைந்த மாதிரி மொழி (Unified Modeling Language) யானது மாதிரியை வரைபடமாக எடுத்துக்கூற உதவுகின்றது.

இங்கு முன்வைக்கப்பட்ட தகவல் மறுநோக்கத்திற்கான தீர்வானது பல்வேறுபட்ட முறைக்கான (Multimedia) மாதிரிக்கான மாதிரியை (Meta Model) அடிப்படையாகக் கொண்டது.

தகவல் மறுநோக்கமானது கீழ்க்கண்டவற்றால் வரையறுக்கப்படுகின்றது. அவையாவன தகவல், குறியீடு, அமைப்பிற்கான தகவலின்மை (Non availability of formatting information).

ஒரு மாற்றும் (transformation) பகுதியானது தகவலை எடுத்துக் கொண்டு தேவைக்கு ஏற்ப, புற கட்டளைகளின் அடிப்படையில் அதன் அமைப்பை மாற்றுகின்றது. இது தகவல்களை மாற்றாது, ஆனால் தேவைக்கு ஏற்ப தகவல்களை கூட்டி அல்லது குறைத்து வெளிப்படுத்தும்.

ஒரு உருவப்படுத்தும் (Imaging) பகுதியானது தேவைக்கு ஏற்ப உருவ,வடிவ அமைப்புகளை (formatting) மாற்றும் வல்லமை உடையது. மேற்சொன்ன இரண்டு பகுதிகளும் அளிப்பானிலோ (server), பெறுவானிலோ (client) செயல்படுத்த முடியும்.

இந்த மாதிரியை அடிப்படையாகக் கொண்ட அனுகுமுறையானது ஓ.எம்.ஜி (O.M.G – Object Management Group)யின் எம்.டி.ஏ (M.D.A – Model Driven Architecture)வால் உந்தப்பட்டது. இது இயக்கத்திற்கான வரை முறைகள், பல் தொகுதிகளுக்கு (Functional Units) இடையேயான உறவுகளை விளக்க (Interoperability) பல்வேறு அடுக்குகளான மாதிரிகளை முன் மொழிகின்றது.

இந்த ஆய்வானது சிறப்பான தகவல் மறுநோக்கத்தின் மேன்மையையும், மாதிரியுடை அனுகூலத்தையும் விளக்க தானியங்கி காசாளர் கருவி (Automated Teller Machine)க்கான பொருள் களஞ்சியத்தில் (Data Dictionary) தகவல் மறுநோக்கத்தை ஏற்படுத்தியுள்ளது. இதை அடைய ஒருங்கிணைந்த மாதிரி மொழி மாதிரிகளை தயார் செய்யவும், தகவல்களை வெளிப்படுத்த விரிவுபடுத்தப்பட்ட குறியீட்டு மொழியும் (Extended Markup Language) உபயோகப்படுத்தப்பட்டன.

## ACKNOWLEDGEMENT

I solemnly bow my head to the **Lord Almighty** for His immense grace at each and every part of my life.

I sincerely thank our Principal **Dr.K.K.Padmanabhan., Ph.D.,** Kumaraguru College of Technology, Coimbatore, for the patronage and innumerable facilities provided by him for the thesis work.

I am profoundly grateful to my guide **Dr.S.Thangasamy,Ph.D.,** Head of the Department, Computer Science and Engineering, Kumaraguru College of Technology, Coimbatore, for his valuable guidance, constant encouragement, monitoring and above all his humanitarian approach towards me throughout the course of my project.

I record my sincere thanks to our project coordinator **Mr.R.Dinesh.M.S.,** Assistant Professor, Department of Computer Science and Engineering, Kumaraguru College of Technology, Coimbatore, for his constant encouragement and motivation.

I express my sincere thanks to our course coordinator **Mrs.L.S.Jayashree,M.E.,** Senior Lecturer, Department of Computer Science and Engineering, Kumaraguru College of Technology, Coimbatore, for her motivation and guidance.

I take this opportunity to extend my sincere and open thanks to my family members for their love, moral and financial help given to me at this age and stage. I extend my sincere thanks to all the hearts that have helped me to successfully complete this course.

## TABLE OF CONTENTS

| <b>CONTENTS</b>  | <b>PAGE No.</b> |
|--|-----------------|
| <b>Abstract</b>  | iii             |
| <b>Acknowledgement</b>                                   | vii             |
| <b>List of figures</b>                                   | x               |
| <br>   |                 |
| <b>CHAPTER 1 INTRODUCTION</b>                            | <b>1</b>        |
| 1.1 Introduction   | 1               |
| 1.2 Applications for content repurposing                 | 2               |
| <br>   |                 |
| <b>CHAPTER 2 LITERATURE REVIEW</b>                       | <b>3</b>        |
| 2.1 Introduction   | 3               |
| 2.1.1 Types of reuse                                     | 3               |
| 2.1.2 Sharing  | 4               |
| 2.1.3 Multipurpose                                       | 4               |
| 2.1.4 Repurpose  | 4               |
| 2.2 Content Repurposing                                  | 5               |
| 2.2.1 Driving forces -Technical                          | 5               |
| 2.2.2 Driving forces - Pedagogical (Science of Teaching) | 5               |
| 2.2.3 Driving forces - General                           | 6               |
| 2.2.4 Over heads due to content repurposing              | 6               |
| 2.2.5 Need for content repurposing                       | 6               |
| 2.3 Existing Techniques                                  | 6               |
| 2.3.1 Bridges  | 6               |
| 2.3.2 Web standards                                      | 7               |
| 2.3.3 Procedural Markup Language                         | 7               |
| 2.3.4 Framework  | 8               |



|                                      |   |           |
|--------------------------------------|---|-----------|
| 2.4                                  | Model Driven Architecture                           | 8         |
| 2.4.1                                | Introduction  | 8         |
| 2.4.2                                | Advantage of models                                 | 9         |
| 2.4.3                                | Model driven approach                               | 9         |
| 2.5                                  | Multimedia Meta model                               | 10        |
| 2.5.1                                | Modules in multimedia metamodel                     | 11        |
| 2.5.2                                | Physical foundations                                | 11        |
| 2.5.3                                | Computing factors                                   | 12        |
| 2.5.4                                | Human factors                                       | 13        |
| 2.5.5                                | Content repurposing use cases                       | 13        |
| <b>CHAPTER 3</b>                     | <b>LINE OF ATTACK</b>                               | <b>14</b> |
| 3.1                                  | Introduction  | 14        |
| 3.2                                  | Data dictionary                                     | 15        |
| 3.3                                  | UML for modeling                                    | 18        |
| 3.4                                  | XML for representation                              | 22        |
| <b>CHAPTER 4</b>                     | <b>DETAILS OF METHODOLOGY EMPLOYED</b>              | <b>24</b> |
| 4.1                                  | Model for ATM system                                | 24        |
| 4.2                                  | Model for data dictionary                           | 38        |
| 4.3                                  | Representation of data dictionary using XML         | 41        |
| <b>CHAPTER 5</b>                     | <b>IMPLEMENTATION RESULTS</b>                       | <b>44</b> |
| 5.1                                  | Result for an entry of a data dictionary - ATM card | 44        |
| 5.2                                  | Advantages due to this approach                     | 48        |
| 5.3                                  | Comparison with RDBMS approach                      | 49        |
| <b>CONCLUSION AND FUTURE OUTLOOK</b> |   | <b>51</b> |
| <b>REFERENCES</b>                    |   | <b>52</b> |

## LIST OF FIGURES

| FIGURE | CAPTION  | PAGE NO |
|--------|--|---------|
| 2.1    | Multimedia metamodel   | 11      |
| 4.1    | Use case for ATM system  | 27      |
| 4.2    | Activity diagram for cash with drawl                               | 28      |
| 4.3    | System start up sequence diagram                                   | 29      |
| 4.4    | System shutdown sequence diagram                                   | 30      |
| 4.5    | Session sequence diagram   | 31      |
| 4.6    | Transaction sequence diagram                                       | 33      |
| 4.7    | User interface diagram for ATM                                     | 35      |
| 4.8    | Class diagram for ATM  | 37      |
| 4.9    | Top level use case view  | 38      |
| 4.10   | Analysis model main diagram  | 39      |
| 4.11   | Browse data dictionary main sequence diagram                       | 39      |
| 4.12   | Search data dictionary main sequence diagram                       | 40      |
| 4.13   | Edit data dictionary sequence diagram                              | 41      |
| 5.1    | XML document for the entry ATM card                                | 45      |
| 5.2    | User interface to retrieve information from the<br>data dictionary | 45      |
| 5.3    | Personalized view for trainee                                      | 46      |
| 5.4    | Personalized view for junior programmer                            | 46      |
| 5.5    | Personalized view for senior programmer                            | 47      |
| 5.6    | Personalized view for group leader                                 | 47      |
| 5.7    | Personalized view for analyst                                      | 48      |

## CHAPTER 1

### INTRODUCTION

#### 1.1 INTRODUCTION

We know that in today's intellectual era information is the asset. Especially due to the magical Internet and Intranet we are having the opportunity to share the information regarding any thing (Content). But it so happens that the content is to be used by different sections of people cutting across language, knowledge level, format, hardware and platform capabilities.

Internet-enabled cell phones, PDAs(Personal Digital Assistant), desktop, laptop, and wearable PCs(Personal Computer) have quite different requirements and presentation capabilities. The problem here is, in order to meet the above said requirements, the content needs to be treated in a special approach, since much online content has been already created by not taking into account the above differences. If we want to access this content efficiently through a broad variety of pervasive -access devices, we need to rethink how it is to be specified and created. Content repurposing can help us to create a broad content base for various device profiles by reusing existing content and adjusting it to fit new situations.

However, taking advantage of content repurposing to its full potential is a challenging task. Since the number of device profiles available for accessing online content can be counted in hundreds. A large variety of formats and standards for online content makes the situation even worse since there is no consensus on how to unify them.

In this report, I have proposed a model-driven framework for content repurposing with the data dictionary of ATM system development. The proposed solution introduces concepts and mechanisms used in the multimedia domain. Using this Multimedia metamodel, I have explored novel design approach for content repurposing associated with the data dictionary of ATM system.

## **1.2 APPLICATIONS FOR CONTENT REPURPOSING**

We can use the technique of content repurposing where ever the information is reused in a network environment. i.e. both internet and intranet. We know that nowadays most of the organizations do have intranet for communication \ sharing \ exchange of information. Many of them have their public information made available through the world wide web. i.e. through a web site of their own. In such a situation, we can apply content repurposing technique to have an automated choice or adaptation to the context like in

- Change of subject knowledge level from high school student to post-secondary students.
- Change of discipline from medical students to nurses
- Change of spelling from American language to Canadian language.
- Change of language from English to French.
- Customize for jurisdiction, company, faculty, or university

## **CHAPTER 2**

### **LITERATURE SURVEY**

#### **2.1 INTRODUCTION**

The natural human tendency is to enjoy comfort with little or no work. When we look into the effort, time and cost spent for developing, updating, maintaining contents in various intranet and internet web contents it will be beneficial if it is possible to reuse the same content cutting across platforms, formats and users. Due to reusability we will get the following advantages:

- Reduction of effort
- Reduction in cost
- Reduction in time

##### **2.1.1 Types Of Reuse**

Based upon the treatment to be carried out on the content regarding platform and the level of audience we can categorize the reuse into three types. They are

- Sharing
- Multipurpose
- Repurposing

### **2.1.2 Sharing**

To use again, with little or no special treatment or processing assumes platform and audience to be the same.

#### **Examples for Sharing**

- Granting rights to all the users to access a specific file
- Exchanging the content through transferring the files between users.

### **2.1.3 Multipurpose**

To use again, especially after special treatment or processing permit reuse across media. This requires explicit archiving and technical standards.

#### **Examples of Multipurpose**

- Analog information available in the videotape is converted to digital information in a CD-ROM.
- Information in a CD is converted to information in the web
- Photoshop file with format as \*.psd (Photoshop data) is converted to a format suitable for web (as jpg Joint Photographic Expert Group format)
- Information in HTML representation for computer is modified for PDA delivery to meet its display characteristics.

### **2.1.4 Repurposing**

To use again, especially after special treatment or processing permitting reuse across media and audiences.

## **2.2 CONTENT REPURPOSING**

It adjusts existing content to enable its reuse for various device profiles and usage. Automatically transforming marked-up content for presentation in multiple applications.

Another way of defining is modifying the content or outlook or both according to user. We will see the driving forces which need to be addressed while doing content repurposing

### **2.2.1 Driving Forces –Technical**

- Change of delivery platform (Content available in the CD-ROM needs to be presented in the Web, Content available in the Web needs to be displayed in Personal Digital Assistant)
- Performance Changes such as increase in bandwidth, decrease in bandwidth, increase / decrease in processor speed.
- Change in the display resolution
- Operating System Changes

### **2.2.2 Driving Forces –Pedagogical (Science of Teaching)**

- Change of subject knowledge level from high school student to post-secondary students.
- Change of discipline from medical students to nurses
- Change of spelling from American language to Canadian language.
- Change of language from English to French.

### **2.2.3 Driving Forces –General**

- Bug fixing
- Feature enhancement
- Branding

### **2.2.4 Over heads due to content repurposing**

Repurposing documents can be difficult and time consuming.

### **2.2.5 Need to do content repurposing**

In order to decide whether we need to have web content repurposing or not, we should ask these questions.

- a) Will my site visitors want this content?
- b) Is online text the best format for this content?
- c) Does the content supports the mission of the site?
- d) Will the content integrate into the existing site structure?(or will I have to change the site structure to accommodate the new content?)
- e) Does this content have a long enough shelf life to make repurposing worthwhile?

## **2.3 EXISTING TECHNIQUES**

### **2.3.1 Bridges**

Authoring tools provide limited content repurposing. They transform their native format to and from other formats. (Export/Import)



*Drawback:-*

These bridges are unsuitable for the highly dynamic on-line world.

Because!

- i) Developing content bridges is a difficult and costly process as bridges require detailed knowledge of proprietary formats and interfaces. If N is the number of target platforms, we need  $N^2-N$  bridges.
- ii) A particular bridge's processing logic is not necessarily reusable in constructing other bridges which greatly increases the development cost.

### **2.3.2 Web Standards**

Standards HTML, DHTML helps to face web publishing challenges.

*Drawback:-*

Cover only a subset of multimedia content space and usage scenarios.

### **2.3.3 Procedural Markup Language**

PML is used to separate content from presentation [10]. It specifies knowledge structures, the underlying physical media and the relationship between them using cognitive media roles. PML can be translated into various presentations depending on the context, goals and user expertise.

*Drawback:-*

Do not directly address the problem of repurposing existing content.

### **2.3.4 Framework**

Framework simplifies multimedia software component development by facilitating reuse of code, design patterns and domain expertise. It lets components dynamically adapt presentation quality to the available resource in heterogeneous environment.

*Drawback:-*

More suitable for conventional web access and data oriented web applications and is not directly applicable to online access for pervasive devices

## **2.4 MODEL DRIVEN ARCHITECTURE**

### **2.4.1 Introduction**

A model is a coherent set of formal elements describing something (for example. system, bank, phone, or train) built for some purpose that is amenable to a particular form of analysis, such as

- Communication of ideas between people and machines
- Completeness checking
- Test case generation & verification
- Viability in terms of indicators such as cost and estimation
- Standards
- Transformation into an implementation

A model need not be complete representation of entire system. It may represent one of the functional units of the system. Often, a graphical model excludes code, UML is such one [1] [2] [3]. Moreover, a model has multiple views, some of which are revealed.

*For example*

We can expose individual collaborating state machines on a state chart diagram, or we can emphasize their collaborations directly using a sequence diagram.

### **2.4.2 Advantage of models**

Model-driven development enables reuse at the domain level, increases quality as models are successively improved, reduces costs by using an automated process, and increases software solutions' longevity[4][5]. In this way, models become assets instead of expenses - quite the business proposition!

Model-driven development automates the transformation of models from one form to another. We express each model, both source and target, in some language. Because modeling is an appropriate formalism to formalize knowledge, we can define modeling language's syntax and semantics building a model of the modeling language -a so called metamodel. (The Greek word meta means "after.") For example, the UML standard is written in UML (the UML metamodel), which raises interesting issues with respect to the precise definition of a language defined in terms of itself.

### **2.4.3 Model Driven Approach**

Other domains have faced such problems and it is useful to analyze their experiences and methods [8] [9]. In software development community, for example, developers exchange data using various tools from different vendors. In this case, the model driven approach lets developers efficiently exchange and transform different data and metadata structures.

The essence of proposed solution is the multimedia metamodel, which introduces concepts and mechanisms from the multimedia domain.

Content repurposing is defined by these elements content, containing markup but no formatting information.

This model driven approach is inspired by OMG's (Object Management Group) MDA (Model Driven Architecture) which facilitates system specification and interoperability using hierarchically organized formal models. Specifically, MDA uses a PIM (Platform Independent Model) and one or more PSM (Platform Specific Model) each describing how the base model is implemented on different platforms.



## 2.5 MULTIMEDIA META MODEL

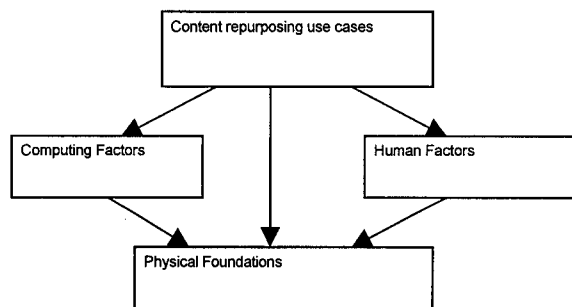
Aim is to identify basic concepts from each multimedia domain and the relation among these concepts [11] [12].

Practical goals are

- A set of precisely defined terms and structured definitions of multimedia concepts.
- High expressiveness to facilitate efficient descriptions
- Knowledge base coherence and interoperability, using standard modeling and storage technologies
- Meta-model scalability to give us the means to define domain concepts at different abstraction levels.

### 2.5.1 Modules in Multimedia Meta model

- Physical foundations
- Computing factors
- Human factors
- Content repurposing use cases



**Figure 2.1 Multimedia Meta Model**

### 2.5.2 Physical Foundations

This introduces concepts for modeling real-world physical properties for use in multimedia presentations.

*Physical media package* has concept definition for stimuli, such as light and sound and their sensible properties, such as intensity, frequency etc.

*Presentation Device package* defines the metamodel for multimedia presentation devices.

*Human Sensory physiology package* contains concept definitions related to the human sensory apparatuses that process stimuli. (We can simplify a data representation by excluding signals that humans are insensitive to, by exploiting effects such as frequency masking or time masking)

### 2.5.3 Computing Factors

Logical foundations

*Presentation dimension* defines information related to a presentation's spatial and temporal dimensions, including coordinate systems and time bases.

*Coding* defines techniques related to content's physical representation (Example-same content in textual or binary encoding)

*Composition* defines mechanisms that seamlessly combine various types of multimedia objects, including temporal composition and spatial composition.

Q.O.S defines ways to quantify a description of platforms at desired quality level.

Presentation platform

It consists of sub packages that define generic descriptions of standard presentation platforms

2D and 3D graphics package

- Textual sub package
- Text to Speech sub package

## 2.5.4 Human factors

Defines various information related to user interactions with online multimedia content.

- Human Perception (Pattern recognition, grouping) & Cognitive Mechanism (memory, context, goals...)
- Social interaction (person, collaborative...)

## 2.5.5 Content repurposing use cases

Use cases provide concepts to describe service functions that support multimedia content operation and management. This module defines mechanisms for content transformation, analysis and personalization; it can make metamodel an active, well integrated part of content repurposing solutions. It consists of 3 packages

- Content transformation

Use case-content transformation provides mechanism for defining content conversions.

- Content analysis

Use case content analysis-defines frame work for content examination approaches (such as data mining)

- Personalization

Use case personalization –defines abstract mechanisms to customize multimedia content based on user profiles

## CHAPTER 3

### LINE OF ATTACK

#### 3.1 INTRODUCTION

To illustrate the content repurposing by model driven architecture approach, I have taken the case of data dictionary of ATM system development. We know that in order to develop an ATM system in a software development environment tens of developers of various levels say trainee, junior programmer, senior programmer, Team leader and Analyst will get involved. They need to have a data dictionary in order to develop the system. We know that data dictionary is a metadata. i.e. data about data. i.e. it contains information of entities involved in the system development. More over this data dictionary is to be shared by all levels of users but with different level of content, based on the user level. More over if any change is to be made in the content of the data dictionary it has to be updated in all the relevant places of the data dictionary.

So to achieve content repurposing using a model driven architecture approach, the line of attack was taken as follows.

- UML diagram models were created for ATM system using Rational Rose tool of Rational Enterprise Edition in order to identify the entries involved in the ATM system development.
- UML diagram models were created for the usage of data dictionary using Smart Draw trail version.



- The data dictionary was represented as a single document using XML.
- User level validation is carried out by java script.
- To display the retrieved data CSS and HTML were used.
- The same functionality (Personalization of the content) is implemented using Oracle 8i as back end and Visual Basic 6 as front end in order to compare the performance with the proposed approach.

### 3.2 DATA DICTIONARY

Data dictionary may cover the whole organization, a part of the organization or a database. In its simplest form, the data dictionary is only a collection of data element definitions, according to descriptions below. More advanced data dictionary contains database schema with reference keys, still more advanced data dictionary contains entity-relationship model of the data elements or objects. The term "data element" is used below. It is the same concept as "data object" or "object" in some database texts.

#### Data element definitions

- *Data element definitions* may be independent of table definitions or a part of each table definition
- *Data element number* is used in the technical documents.
- *Data element name (caption)* commonly agreed, unique data element name from the application domain. This is the real life name of this data element.
- *Short description* -Description of the element in the application domain.

- *Security classification of the data element* - Organization-specific security classification level or possible restrictions on use. This may contain technical links to security systems.
- *Related data elements* - List of closely related data element names when the relation is important.
- *Field name(s)* are the names used for this element in computer programs and database schemas. These are the technical names, often limited by the programming languages and systems.
- *Code format* - Data type (characters, numeric, etc.), size and, if needed, special representation. Common programming language notation, input masks, etc. can be used.
- *Null value allowed* -Null or non-existing data value may be or may not be allowed for an element. Element with possible null values needs special considerations in reports and may cause problems, if used as a key.
- *Default value* - Data element may have a default value. Default value may be a variable, like current date and time of the day.
- *Element coding* (allowed values) and intra-element validation details or reference to other documents
- *Explanation of coding* (code tables, etc.) and validation rules when validating this element alone in the application domain.
- Inter-element validation details or reference to other documents  
Validation rules between this element and other elements in the data dictionary.
- *Database table references* - Reference to tables the element is used and the role of the element in each table. Special indication when the data element is the key for the table or a part of the key.
- *Definitions and references* needed to understand the meaning of the element.

- *Short application domain definitions and references* to other documents needed to understand the meaning and use of the data element.
- *Source of the data in the element* -Short description in application domain terms, where the data is coming. Rules used in calculations producing the element values are usually written here.
- *Validity dates for the data element definition* - start and possible end dates, when the element is or was used. There may be several time periods the element has been used.
- *History references*-Date when the element was defined in present form, references to supersede elements, etc.
- *External references* - References to books, other documents, laws, etc.
- *Version of the data element document* -Version number or other indicator. This may include formal version control or configuration management references, but such references may be hidden, depending on the system used.
- *Date of the data element document* - writing date of this version of the data element document.
- *Quality control references Organization* – specific quality control endorsements, dates, etc.
- *Data element notes* - Short notes not included in above parts.

#### Table definitions (For representing the data dictionary using RDBMS)

- Table definition is usually available with SQL command help table
- Table name
- Table owner or database name
- List of data element (column) names and details
- Key order for all the elements, which are possible keys

- Possible information on indexes
- Possible information on table organization  
Technical table organization, like hash, heap, B+ -tree, AVL -tree, ISAM, etc. may be in the table definition.
- Duplicate rows allowed or not allowed
- Possible detailed data element list with complete data element definitions
- Possible data on the current contents of the table  
The size of the table and similar site specific information may be kept with the table definition.
- *Security classification of the table* is usually same or higher than its elements. However, there may be views accessing parts of the table with lower security.

*Database schema* is usually graphical presentation of the whole database. Tables are connected with external keys and key columns. When accessing data from several tables, database schema will be needed in order to find joining data elements and in complex cases to find proper intermediate tables. Some database products use the schema to join the tables automatically.

### **3.3 UML FOR MODELING**

*OMG* - The Object Management Group is an international, not-for-profit industrial consortium that creates and maintains software interoperability specifications. The OMG's specifications include the UML modeling notation, XMI (XML metadata interchange), CORBA (common object request broker architecture) middleware, and dozens of domain-specific interoperability specifications in such areas as transportation, life sciences, telecommunications, and manufacturing.

The Unified Modeling Language is an industry standard visual language for modeling software systems. These models capture knowledge about a system at various abstraction levels, ranging from requirements and analysis models to design models. This means that modelers can specify software systems using higher-level domain-oriented concepts that abstract away much of the underlying implementation technology used to realize such systems. UML enables automated tools that interchange and transform models as part of the process and generate a system's implementation artifacts (typically source code and metadata).

The heart of object-oriented problem solving is the construction of a model. The model abstracts the essential details of the underlying problem from its usually complicated real world. Several modeling tools are wrapped under the heading of the UML™, which stands for Unified Modeling Language™.

At the center of the UML are its nine kinds of modeling diagrams, which is described here.

- Use case diagrams
- Class diagrams
- Object diagrams
- Sequence diagrams
- Collaboration diagrams
- State chart diagrams
- Activity diagrams
- Component diagrams
- Deployment diagrams

## Importance of UML

Writing software is not like constructing a building. The more complicated the underlying system, the more critical the communication among everyone involved in creating and deploying the software. In the past decade, the UML has emerged as the software blueprint language for analysts, designers, and programmers alike. It is now part of the software trade. The UML gives everyone from business analyst to designer to programmer a common vocabulary to talk about software design

The UML is applicable to object-oriented problem solving. Anyone interested in learning UML must be familiar with the underlying tenet of object-oriented problem solving -- it all begins with the construction of a model. A model is an abstraction of the underlying problem. The domain is the actual world from which the problem comes.

Models consist of objects that interact by sending each other message. Think of an object as "alive." Objects have things they know (attributes) and things they can do (behaviors or operations). The values of an object's attributes determine its state.

Classes are the "blueprints" for objects. A class wraps attributes (data) and behaviors (methods or functions) into a single distinct entity. Objects are instances of classes.

## Use case diagrams

- Use case diagrams describe what a system does from the standpoint of an external observer. The emphasis is on what a system does rather than how.
- Use case diagrams are closely connected to scenarios. A scenario is an example of what happens when someone interacts with the system.
- A use case is a summary of scenarios for a single task or goal. An actor is who or what initiates the events involved in that task. Actors are simply roles that people or objects play.
- A use case diagram is a collection of actors, use cases, and their communications.
- Use case diagrams are helpful in three areas.
- Determining features (requirements). New use cases often generate new requirements as the system is analyzed and the design takes shape.
- Communicating with clients. Their notational simplicity makes use case diagrams a good way for developers to communicate with clients.
- Generating test cases. The collection of scenarios for a use case may suggest a suite of test cases for those scenarios.

*Class diagrams* - A Class diagram gives an overview of a system by showing its classes and the relationships among them. Class diagrams are static -- they display what interacts but not what happens when they do interact

*Object- interaction diagram* - It depicts the messages and message arguments that objects send to one another. This comes in two styles

*Sequence Diagram* - This emphasizes temporal sequence over static object relationships.

*Collaboration Diagram* - It gives static object relationships and dynamic object messaging.

*Activity Diagram* - An activity diagram is a variation or a special case of state machine, in which the states are activities representing the performance of operations and the transitions are triggered by the completion of operations. The purpose of an activity diagram is to provide a view of flows and what is going on inside a use case or among several classes.

### **3.4 XML FOR REPRESENTATION**

*XML (EXTENSIBLE MARKUP LANGUAGE)* is a system for defining, validating and sharing document formats? It uses tags to distinguish document structures and attributes to encode extra document information [7]. It looks/works very similar to SGML.

*Advantages of XML:*

- i) System independent, vendor independent approach to document interchange over the web
- ii) It can distribute a significant proportion of the processing load from the web server to the web client.



- iii) It can present different views of the same data. Users can switch between views without requiring that the data to be downloaded again from the server in a different form.
- iv) Intelligent web agents tailor information discovery to individual users.
- v) It is a license-free, platform-independent technology (a W3C technology)
- vi) It provides reliable data structure that any application can read and parse. This data can then be easily transformed from one XML schema to another.
- vii) It provides simple way to share between computer systems from multiple companies.
- viii) It is Portable.
- ix) It can be used to design new markup languages (e.g., MathML)
- x) It is extensible (creating their own tags)
- xi) It provides structure (can represent database schemas or Object oriented hierarchies).
- xii) It can be used for validation. (Parse the grammar it uses to define data to an outside application).

## CHAPTER 4

### DETAILS OF METHODOLOGY EMPLOYED

#### 4.1 MODEL FOR ATM SYSTEM

##### *Requirements statement for ATM System (Automated Teller Machine)*

The software to be designed will control a automated teller machine (ATM) having a magnetic stripe reader for reading an ATM card, a customer console (keyboard and display) for interaction with the customer, a slot for depositing envelopes, a dispenser for cash (in multiples of Rs.100.00), a printer for printing customer receipts, and a key-operated switch to allow an operator to start or stop the machine. The ATM will communicate with the bank's computer over an appropriate communication link. (The software on the latter is not part of the requirements for this problem.)

The ATM will service one customer at a time. A customer will be required to insert an ATM card and enter a personal identification number (PIN) - both of which will be sent to the bank for validation as part of each transaction. The customer will then be able to perform one or more transactions. The card will be retained in the machine until the customer indicates that he/she desires no further transactions, at which point it will be returned - except as noted below.

The ATM must be able to provide the following services to the customer:

- A customer must be able to make a cash withdrawal from any suitable account linked to the card, in multiples of Rs.100.00. Approval must be obtained from the bank before cash is dispensed.
- A customer must be able to make a deposit to any account linked to the card, consisting of cash and/or checks in an envelope. The customer will enter the amount of the deposit into the ATM, subject to manual verification when the envelope is removed from the machine by an operator. Approval must be obtained from the bank before physically accepting the envelope.
- A customer must be able to make a transfer of money between any two accounts linked to the card.
- A customer must be able to make a balance inquiry of any account linked to the card.
- A customer must be able to abort a transaction in progress by pressing the Cancel key instead of responding to a request from the machine.
- The ATM will communicate each transaction to the bank and obtain verification that it was allowed by the bank. Ordinarily, a transaction will be considered complete by the bank once it has been approved. In the case of a deposit, a second message will be sent to the bank indicating that the customer has deposited the envelope. (If the customer fails to deposit the envelope within the timeout period, or presses cancel instead, no second message will be sent to the bank and the deposit will not be credited to the customer.)
- If the bank determines that the customer's PIN is invalid, the customer will be required to re-enter the PIN before a

transaction can proceed. If the customer is unable to successfully enter the PIN after three tries, the card will be permanently retained by the machine, and the customer will have to contact the bank to get it back.

- If a transaction fails for any reason other than an invalid PIN, the ATM will display an explanation of the problem, and will then ask the customer whether he/she wants to do another transaction.
- The ATM will provide the customer with a printed receipt for each successful transaction, showing the date, time, machine location, type of transaction, account(s), amount, and ending and available balance(s) of the affected account ("to" account for transfers).
- The ATM will have a key-operated switch that will allow an operator to start and stop the servicing of customers. After turning the switch to the "on" position, the operator will be required to verify and enter the total cash on hand. The machine can only be turned off when it is not servicing a customer. When the switch is moved to the "off" position, the machine will shut down, so that the operator may remove deposit envelopes and reload the machine with cash, blank receipts, etc.
- The ATM will also maintain an internal log of transactions to facilitate resolving ambiguities arising from a hardware failure in the middle of a transaction. Entries will be made in the log when the ATM is started up and shut down, for each message sent to the Bank (along with the response back, if one is expected), for the dispensing of cash, and for the receiving of an envelope. Log entries may contain card numbers and amounts, but for security will *never* contain a PIN.

The following UML models for ATM system have been developed in order to identify entries for its data dictionary.

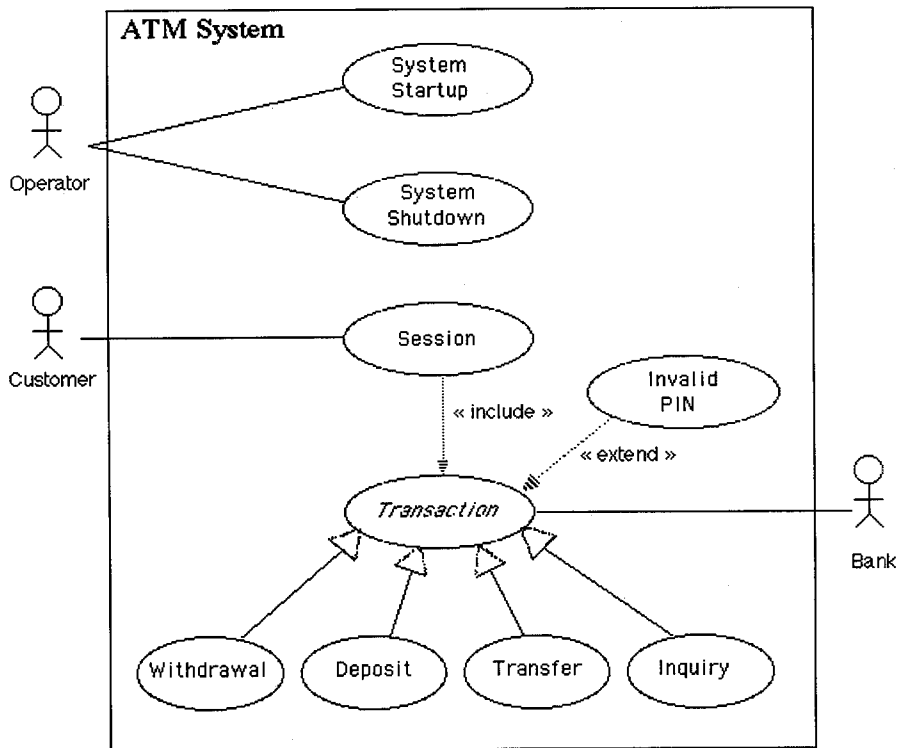


Figure 4.1 Use Case for ATM System

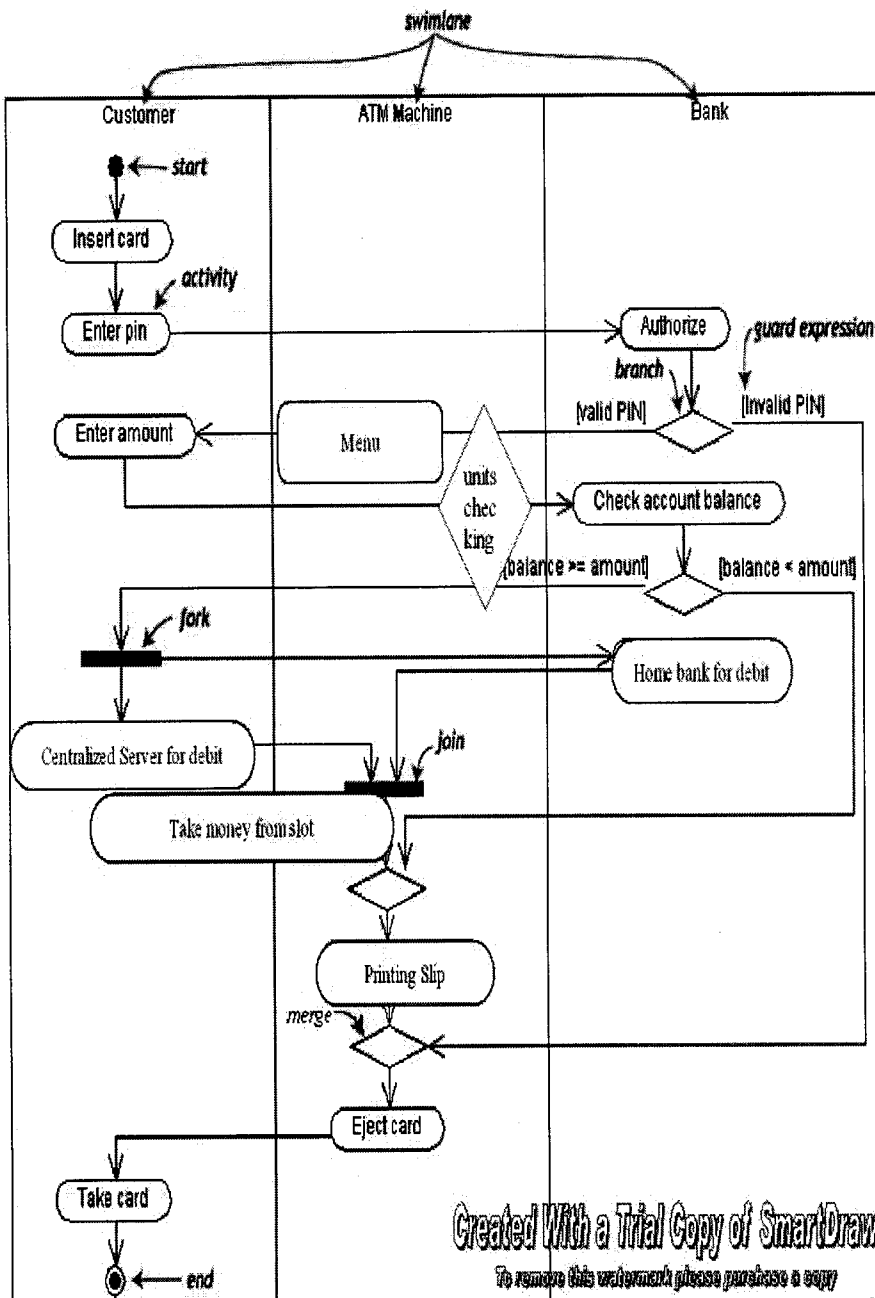


Figure 4.2 Activity diagram for cash with draw

### System Startup Use Case

The system is started up when the operator turns the operator switch to the "on" position. The operator will be asked to enter the amount of money currently in the cash dispenser, and a connection to the bank will be established. Then the servicing of customers can begin.

### System Startup Sequence Diagram

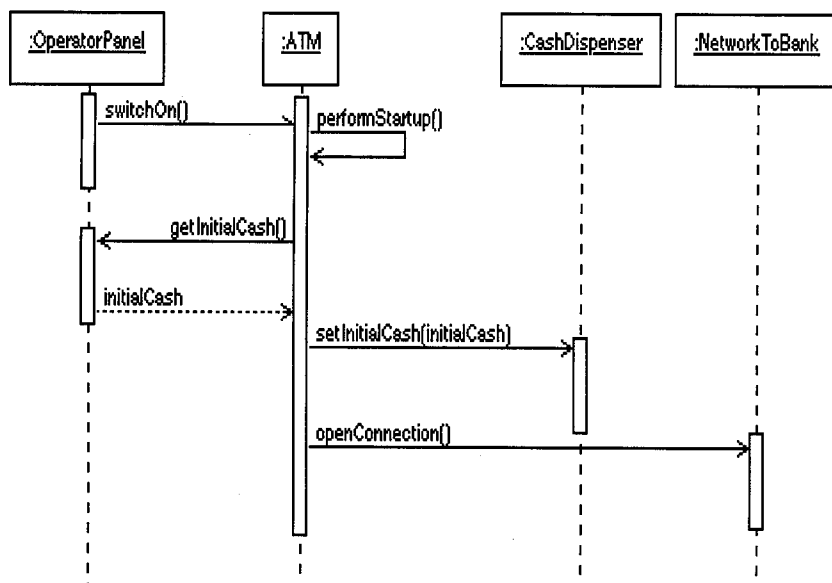


Figure 4.3 System startup sequence diagram

### System Shutdown Use Case

The system is shut down when the operator makes sure that no customer is using the machine, and then turns the operator switch to the "off" position. The connection to the bank will be shut down. Then the operator is free to remove deposited envelopes, replenish cash and paper, etc.

### System Shutdown Sequence Diagram

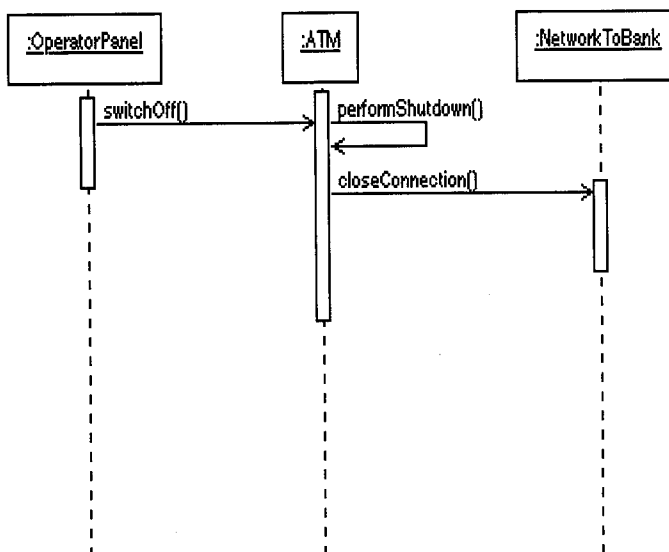


Figure 4.4 System shutdown sequence diagram



## Session Use Case

A session is started when a customer inserts an ATM card into the card reader slot of the machine. The ATM pulls the card into the machine and reads it. (If the reader cannot read the card due to improper insertion or a damaged stripe, the card is ejected, an error screen is displayed, and the session is aborted.) The customer is asked to enter his/her PIN, and is then allowed to perform one or more transactions, choosing from a menu of possible types of transaction in each case. After each transaction, the customer is asked whether he/she would like to perform another. When the customer is through performing transactions, the card is ejected from the machine and the session ends. If a transaction is aborted due to too many invalid PIN entries, the session is also aborted, with the card being retained in the machine. The customer may abort the session by pressing the Cancel key when entering a PIN or choosing a transaction type.

Session Sequence Diagram

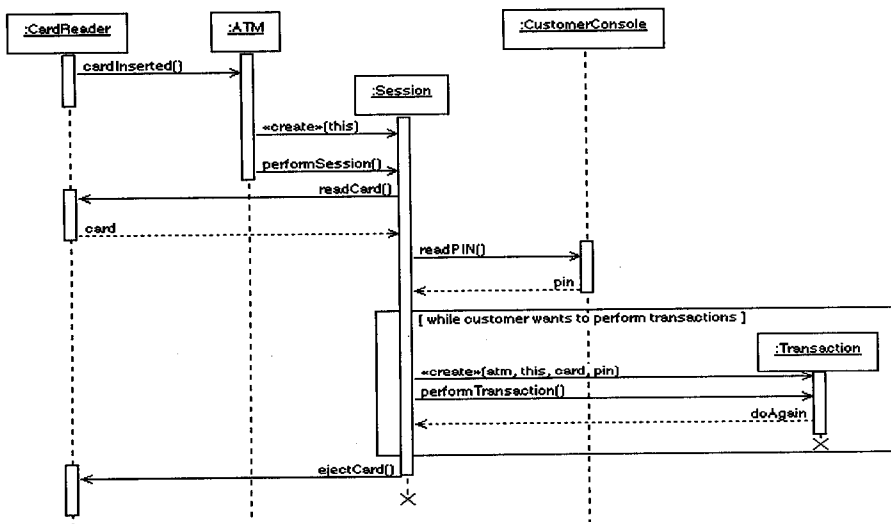


Figure 4.5 session sequence diagram

### *Transaction Use Case*

Transaction is an abstract generalization. Each specific concrete type of transaction implements certain operations in the appropriate way. The flow of events given here describes the behavior common to all types of transaction. The flows of events for the individual types of transaction (withdrawal, deposit, transfer, inquiry) give the features that are specific to that type of transaction.

A transaction use case is started within a session when the customer chooses a transaction type from a menu of options. The customer will be asked to furnish appropriate details (e.g. account(s) involved, amount). The transaction will then be sent to the bank, along with information from the customer's card and the PIN the customer entered.

If the bank approves the transaction, any steps needed to complete the transaction (e.g. dispensing cash or accepting an envelope) will be performed, and then a receipt will be printed. Then the customer will be asked whether he/she wishes to do another transaction.

If the bank reports that the customer's PIN is invalid, the Invalid PIN extension will be performed and then an attempt will be made to continue the transaction. If the customer's card is retained due to too many invalid PIN, the transaction will be aborted, and the customer will not be offered the option of doing another.

If a transaction is cancelled by the customer, or fails for any reason other than repeated entries of an invalid PIN, a screen will be displayed informing the customer of the reason for the failure of the

transaction, and then the customer will be offered the opportunity to do another.

The customer may cancel a transaction by pressing the Cancel key as described for each individual type of transaction below. All messages to the bank and responses back are recorded in the ATM's log.

### Transaction Sequence Diagram

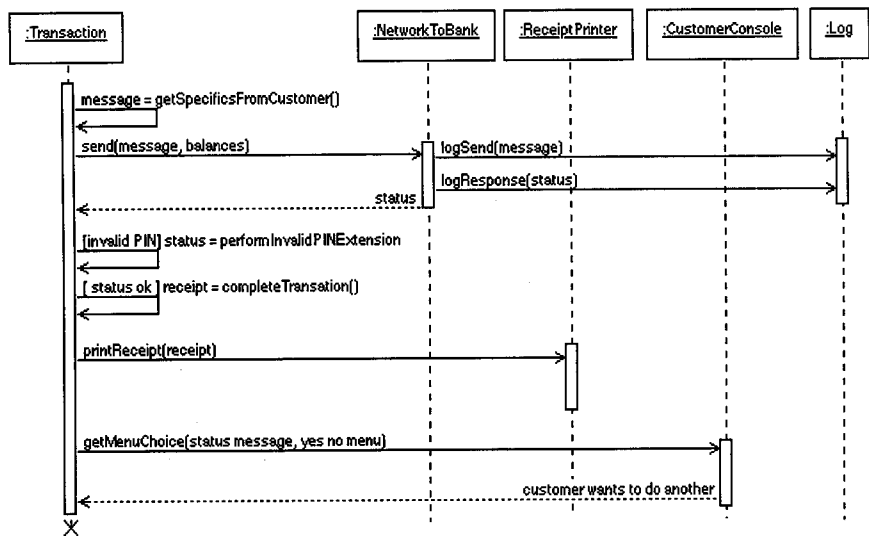


Figure 4.6 Transaction sequence diagram

### *Analysis- Classes*

An initial reading of the use cases suggests that the following will be part of the system.

- A controller object representing the ATM itself (managing the boundary objects listed below.)
- Boundary objects representing the individual component parts of the ATM:
  - Operator panel.
  - Card reader.
  - Customer console, consisting of a display and keyboard.
  - Network connection to the bank.
  - Cash dispenser.
  - Envelope acceptor.
  - Receipt printer.

Controller objects corresponding to use cases. (Note: class ATM can handle the Startup and Shutdown use cases itself, so these do not give rise to separate objects here.)  
Session

Transaction (abstract generalization, responsible for common features, with concrete specializations responsible for type-specific portions). An entity object representing the information encoded on the ATM card inserted by customer. An entity object representing the log of transactions maintained by the machine.

This leads to the following diagram of analysis classes:

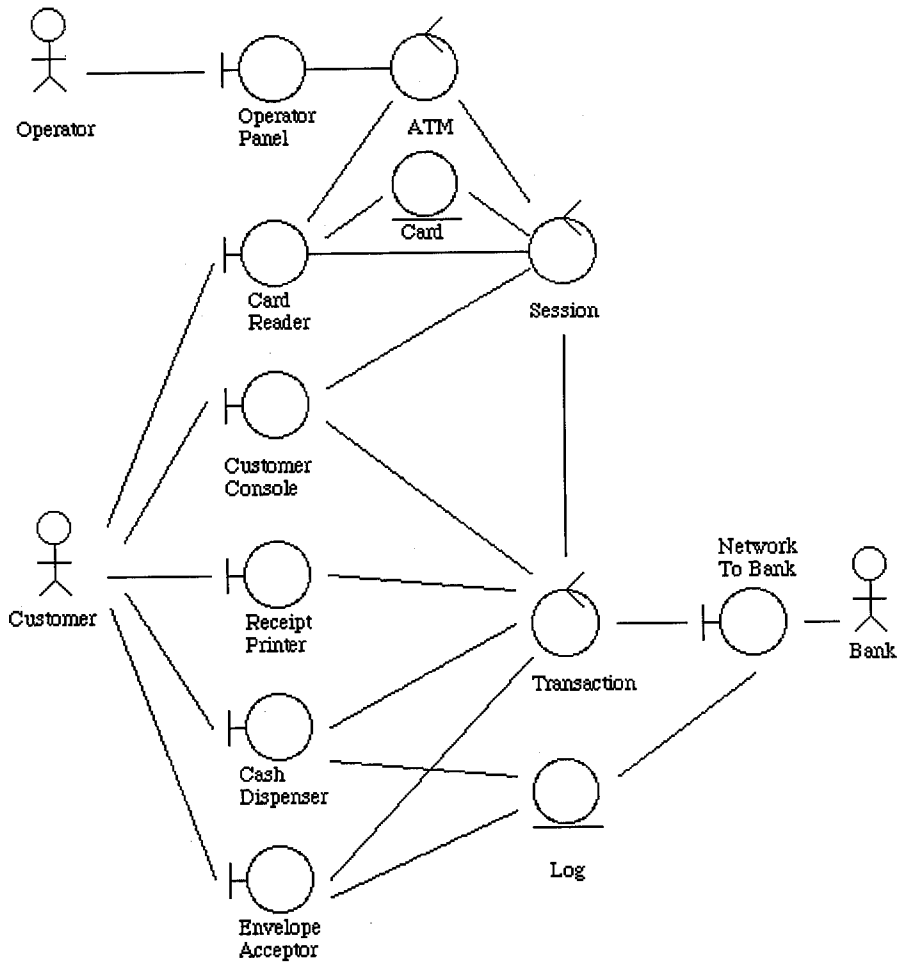


Figure 4.7 User Interface Diagram for ATM

### *Class Diagram for ATM System*

Shown below is the class diagram for the ATM system. The basic structure of the class diagram arises from the responsibilities and relationships discovered when doing Interaction diagrams. (If a class uses another class as a collaborator, or sends a message to an object of that class during an Interaction, then there must either be an association linking objects of those classes, or linking the "sending" class to an object which provides access to an object of the "receiving" class.)

In the case of the ATM system, one of the responsibilities of the ATM is to provide access to its component parts for session and transaction objects; thus, session and transaction have associations to ATM, which in turn has associations to the classes representing the individual component parts. (Explicit "uses" links between session and transaction, on the one hand, and the component parts of the ATM, on the other hand, have been omitted from the diagram to avoid making it excessively cluttered.)

The need for the various classes in the diagram was discovered at various points in the design process.

*Message* - used to represent a message to the bank.

*Receipt* - used to encapsulate information to be printed on a receipt.

*Status* - used to represent return value from message to the bank.

*Balances* - used to record balance information returned by the bank.

*Money* - used to represent money amounts, in numerous places.

*Account Information* - contains names of various types of accounts customer can choose from. That is, OO design is not a "waterfall"

process - discoveries made when doing detailed design and coding can impact overall system design.

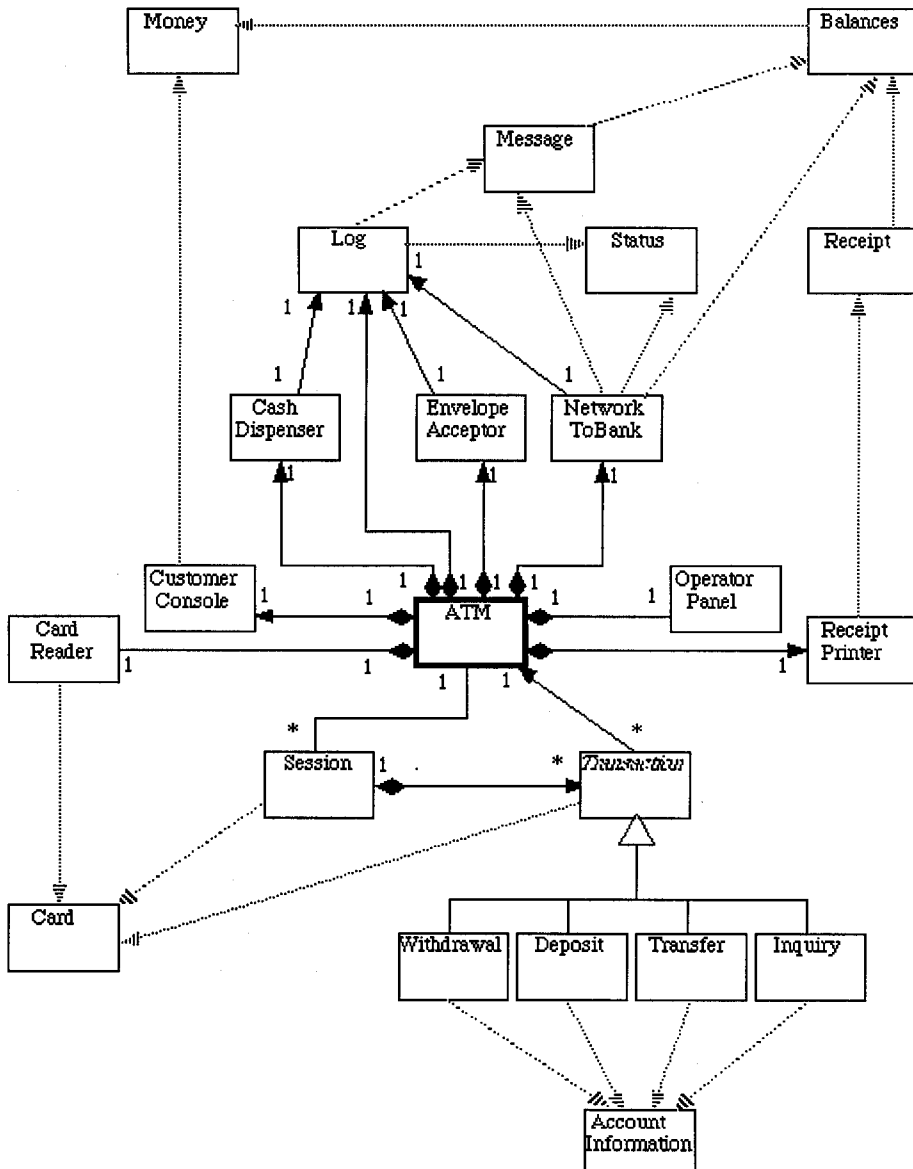


Figure 4.8 Class Diagram

## 4.2 MODEL FOR DATA DICTIONARY

The following models illustrate the data dictionary for ATM system.

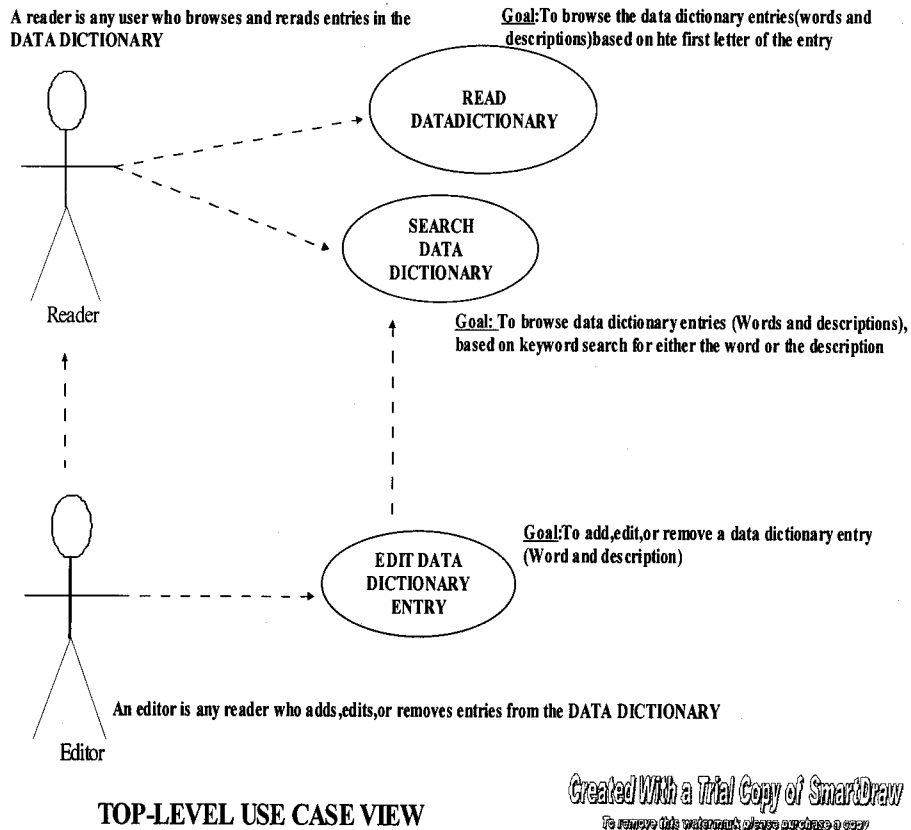
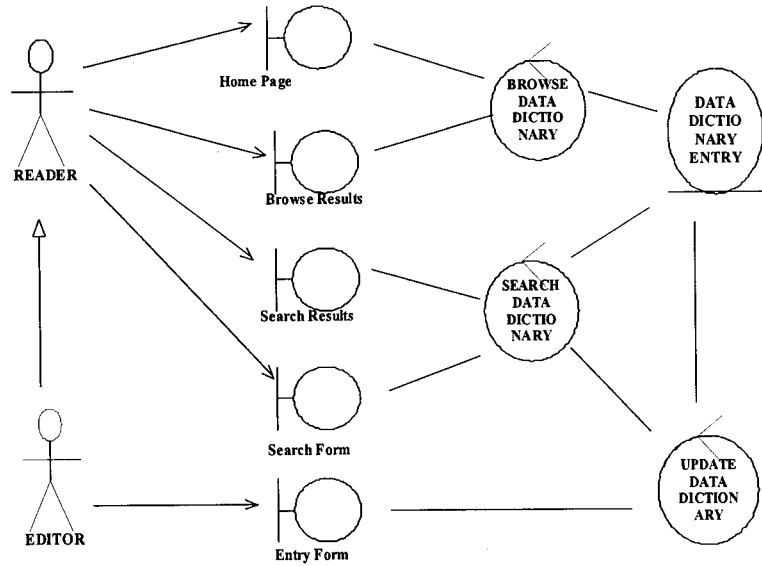


Figure 4.9 Top level use case view

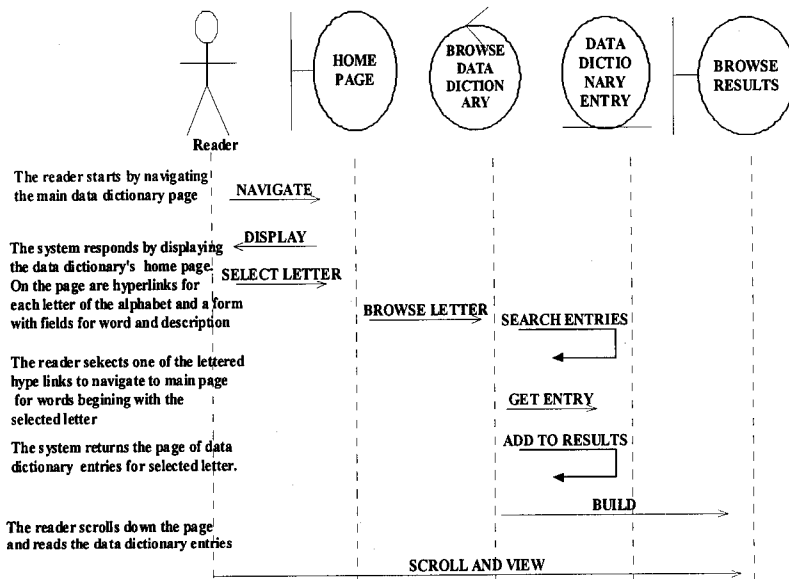




ANALYSIS MODEL: MAIN DIAGRAM

Created With a Trial Copy of SmartDraw  
To remove this watermark please purchase a copy

Figure 4.10 Analysis model main diagram



BROWSE DATA DICTIONARY MAIN SEQUENCE DIAGRAM

Created With a Trial Copy of SmartDraw  
To remove this watermark please purchase a copy

Figure 4.11 Browse data dictionary main sequence diagram

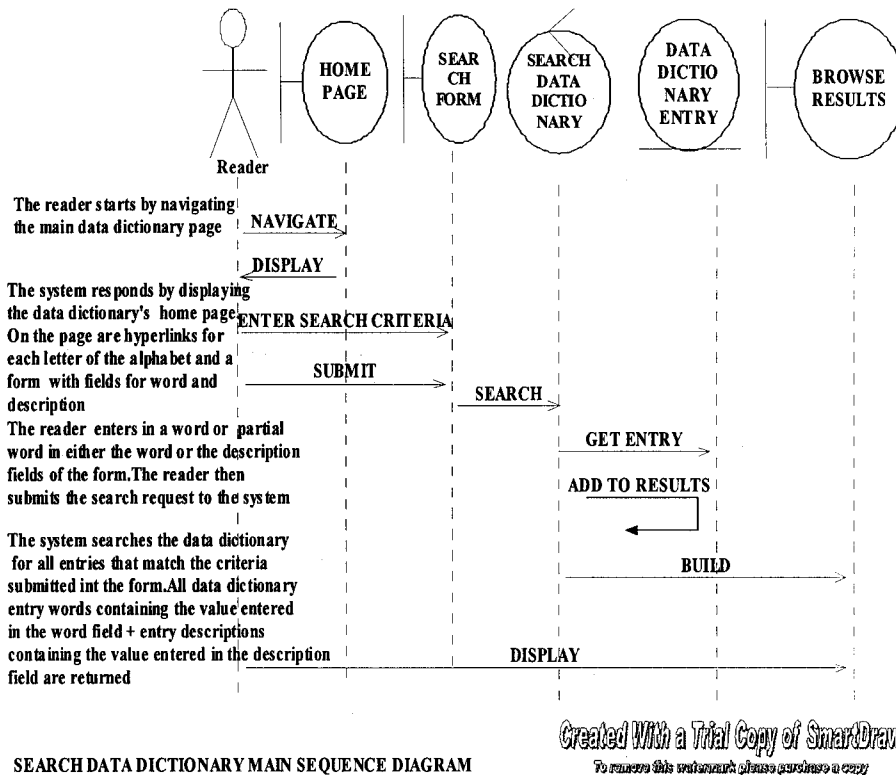


Figure 4.12 Search data dictionary main sequence diagram

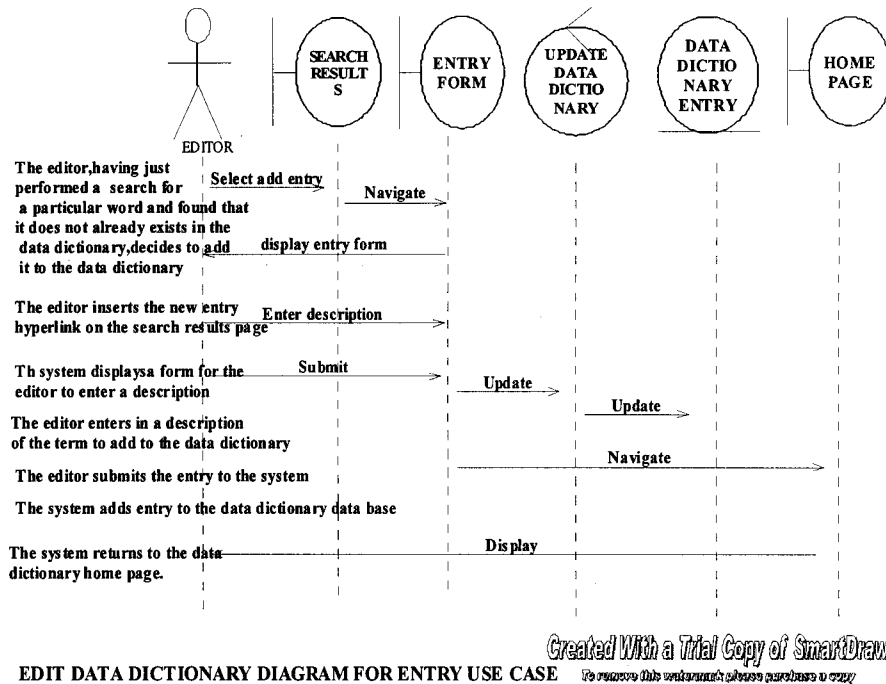


Figure 4.13 Edit data dictionary sequence diagram

### 4.3 REPRESENTATION OF DATA DICTIONARY USING XML

Information was represented using XML in the form of Card and Deck (Playing Cards) arrangement. Depending upon the user level, the content has to be personalized by retrieving the appropriate cards.

#### CARD -1

Key word: ATM card

Description: A plastic card which is used to authenticate the customer based upon the information encoded in it. A card reader used to read and pass data to the validating section. Based upon the validity the customer can do transactions in the ATM system.

## CARD - 2

Information's present on the card with out encoding

- \* Name of the bank
- \* Card holder Name
- \* Card Number
- \* Validity period
- \* Other general instructions and address

## CARD - 3

Information's present in the magnetic stripe in encoded format

- \* Name
- \* Account Number
- \* Type of account
- \* Card Number
- \* Branch in which the customer belongs to

## CARD - 4

Data types used in the magnetic stripe (Assume DB as Oracle)

- \* Name - Varchar(20)
- \* Account Number - Varchar(15)
- \* Card Number - Varchar(10)
- \* Validity Period - Date.

## CARD - 5

Module involved with the card reader:

- \* Validation
- \* Log file in the ATM system

Depending upon the level of the user we can allow the specific cards which are in a single document to be displayed to him. For example five levels of user has been defined, they are Trainee, Junior Programmer, Senior Programmer, Group Leader and Analyst. For example they need to view card 1 – card 1, 2 – card 1, 2, 3 – card 1, 2, 3, 4 - card 1, 2, 3, 4, 5(The sequence can also be random) respectively for the content to get personalized.

As described above the contents of the cards are represented using XML as a single document. All formatting information of the content was represented using CSS (Cascade Style Sheet).

The level of the user is identified with the help of java scripting. Based upon the level of the user, specific cards are retrieved from the single XML document. In order to facilitate this retrieval, XML provides specific feature, which helps to retrieve part of the content from a single document. Thus the retrieved cards are displayed in a HTML page, which also works as a user interface to the user to specify his user name and password. This was demonstrated for the entry 'ATM card' to the data dictionary of ATM system.

## CHAPTER 5

### IMPLEMENTATION RESULTS

#### 5.1 RESULTS FOR AN ENTRY OF A DATA DICTIONARY-(ATM CARD).

The following results were obtained due to personalization of the data dictionary.

- Single document is enough to have all the information corresponding to all levels of users.
- Due to this it reduces number of documents to be created, stored and maintained.
- Reduces storage space requirement and decreases the complexity of having too many documents.
- Inconsistency of information which may arise during updating is avoided since the information will be in only one document.

The following snapshots visualize the personalization of the content.

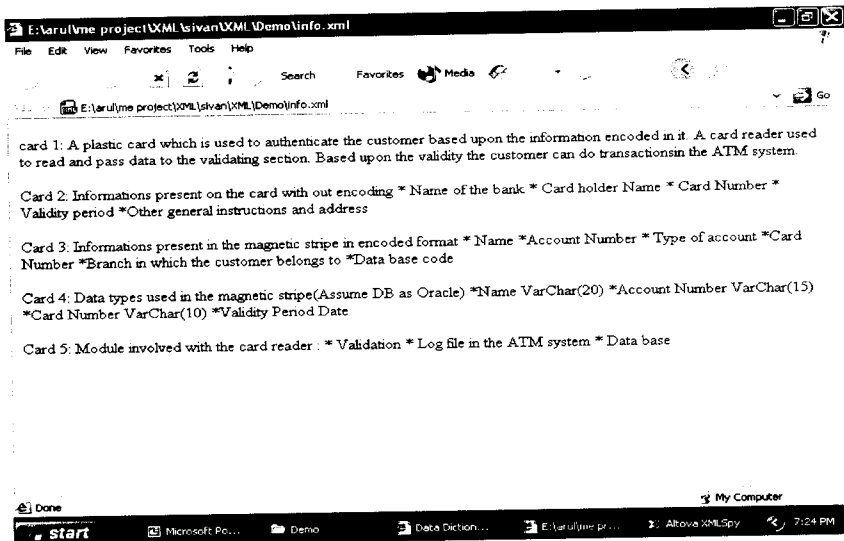


Figure 5.1 XML document for the entry ATM card

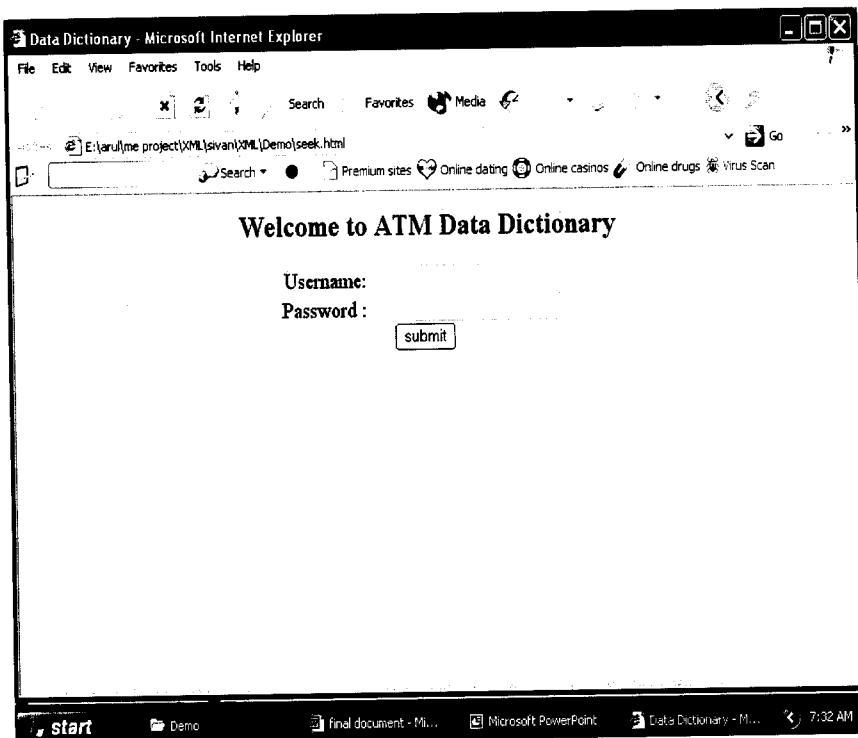


Figure 5.2 User interface to retrieve information from the data dictionary

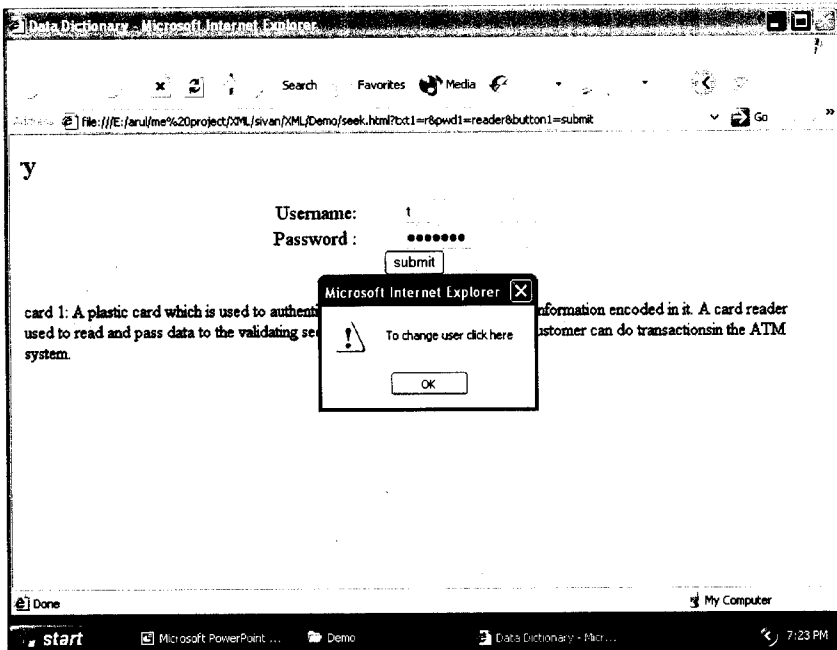


Figure 5.3 Personalized view for trainee

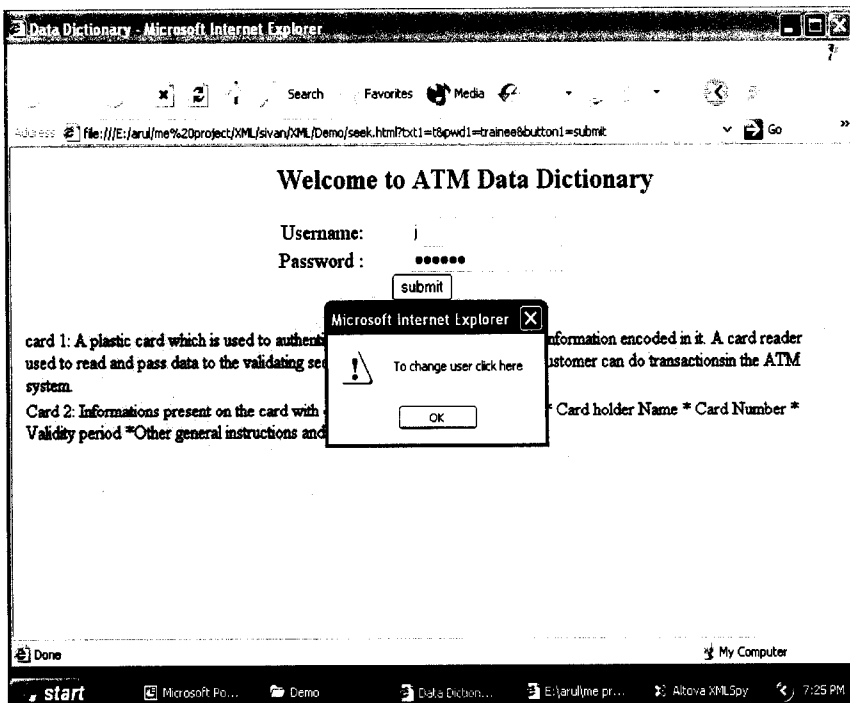


Figure 5.4 Personalized view for junior programmer



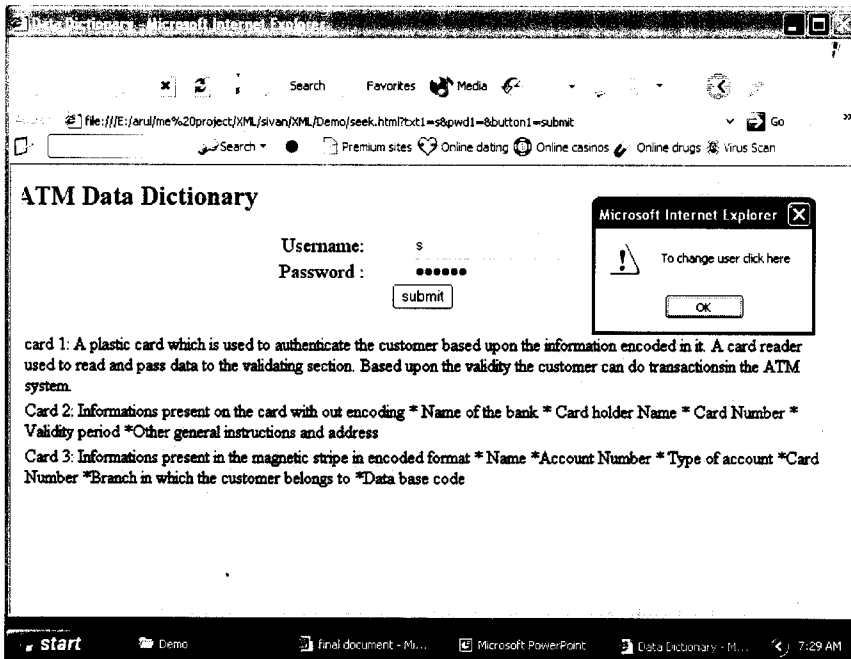


Figure 5.5 Personalized view for senior programmer

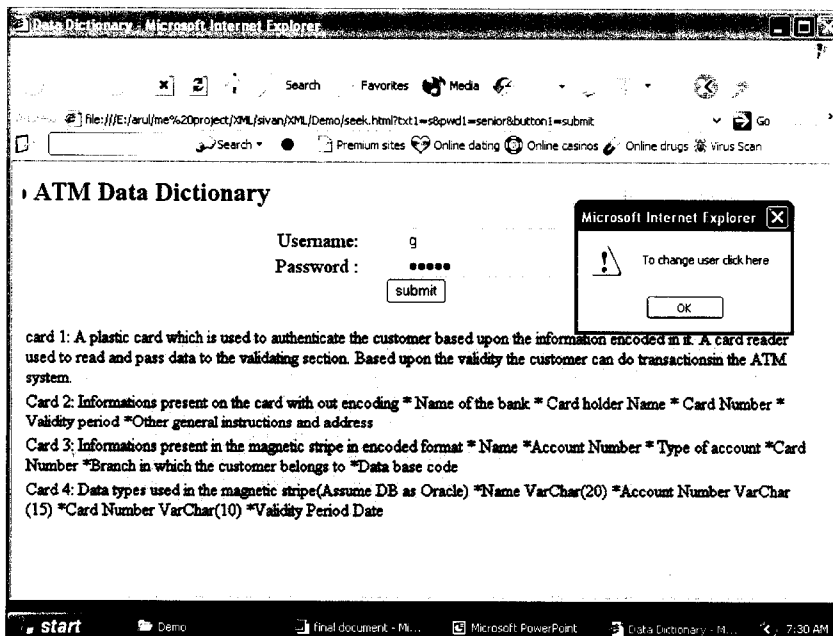


Figure 5.6 Personalized view for group leader

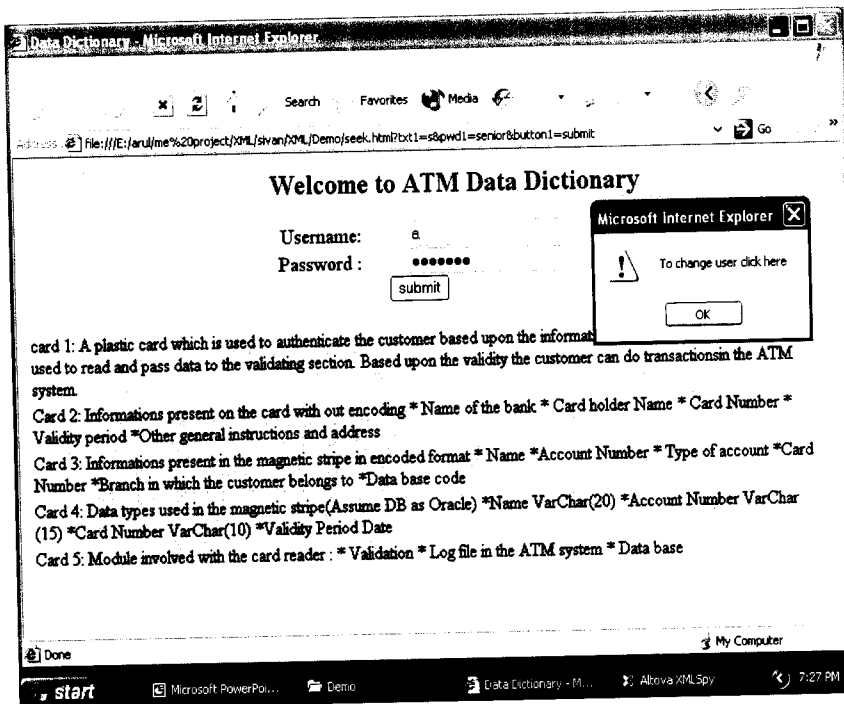


Figure 5.7 Personalized view for analyst

## 5.2 ADVANTAGES DUE TO THIS APPROACH

Some of the major advantages due to this approach are

- Since models are used to develop the system, any change in the system requirement at any stage of system development can be easily incorporated.
- The functionality of the system will be unambiguously understood by all levels of users.
- Due to XML representation the information can be displayed in any HTML browser such as Internet Explorer, which will be convenient to use it in intranet / internet.

More over it offers all the advantages as listed below. Such as system independent, vendor independent approach to document interchanges over the web. It can distribute a significant proportion of the processing load from the web server to the web client. It can present different views of the same data. Users can switch between views without requiring that the data to be downloaded again from the server in a different form. Intelligent web agents tailor information discovery to individual users. It is a license-free, platform-independent technology (a W3C technology). It provides reliable data structure that any application can read and parse. This data can then be easily transformed from one XML schema to another. It provides simple way to share between computer systems of multiple companies.

### **5.3 COMPARISON WITH RDBMS APPROACH**

Using an RDBMS (Example Oracle 8i), same information for the entry 'ATM card' is stored in one table. Another table is used for the same 5 different levels of users as mentioned earlier. The validation and retrieval of appropriate block of information based upon the user is done using a front end tool (Visual Basic 6). The time needed to retrieve specific information based upon the user level is computed. Though much of the difference is not found, it is evident that as the number of tables to be referred increases and as the complexity of the query increase we will be able to find that RDBMS approach will consume more time.

Other than the time factor if we see the maintenance of the data dictionary in RDBMS approach we need to maintain so many tables, more over the advantages of XML representation cannot be achieved. Due to the above factors it is evident that the approach for content repurposing using models and XML representation stands an efficient alternative to other approaches.

## CONCLUSION AND FUTURE OUTLOOK

Due to personalization of the content, we are able to get the benefit of content repurposing (reuse) which reduces cost, time and effort involved in the data dictionary development and maintenance and ensuring access privileges. When compared to the RDBMS approach, the model driven together with XML approach proved to be efficient with respect to cost, time and effort.

More over due to the model driven architecture development, if in future, the customer (bank) requires any change in the ATM system usage such as adding new facilities or functionalities, it can be easily implemented with the help of models.

The following extensions can also be made

- Apart from text data other multimedia data can be used for personalization
- Apart from personalization the following can be achieved
  - Discipline(Education to Entertainment)
  - Level(School level to Research level)
  - Regionalization (spelling U.S to U.K)
  - Language(English to Tamil)

## REFERENCES

1. Booch, Jacobson, Rumbaugh(1999) 'Developing software with UML Object-Oriented Analysis and Design in practice' - Addition-Wesley.
2. Booch, Jacobson, Rumbaugh (1999) 'Fundamentals of Object-Oriented Design in UML' - Addition-Wesley.
3. Booch, Jacobson, Rumbaugh (1999) 'Building web applications with UML' -Addition-Wesley.
4. Bran Selic.(2000) 'A Generic Framework for Modeling Resources with UML' - Computer, IEEE CS Press, pp.64-69.
5. Gary Cernosek, Eric Naiburg (2004) 'The Value of Modeling' A technical discussion of software modeling - IBM Software Group.
6. Gonzales R. (2000) 'Disciplining Multimedia' - IEEE Multimedia, pp. 72-78.
7. Lie H.W. and Saarela J. (1999) 'Multipurpose Web Publishing Using HTML, XML, and CSS' – Communication of the ACM , ACM Press, Vol. 42, No. 10.
8. Mellor S.J. and et al (2003) 'Model Driven Development' - IEEE Software, vol.20, no5, pp14-18.

9. Piero Fraternali and Paolo Paolini Politecnico di Milano (2000) 'Model-Driven Development of Web Applications: The Autoweb System' - ACM Transactions on Information Systems, Vol. 28, No. 4, Pages 323–382.
10. Ram A. and et al. (1999) 'PML: Adding Flexibility to Multimedia Presentations' - IEEE Multimedia, IEEE CS Press, pp. 40-52.
11. Selic B. (2000) 'A generic framework for Modelling resources with UML' - IEEE Computer, vol.33, no.6, pp64-69.
12. Zelijko and Bran Selic (2004) 'A Model driven approach to content repurposing' - IEEE MultiMedia, pp63-71.

