

# User Preference Mining for Context-Aware m-Services Applications

Edgar Jembere

(20047171)

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

Masters of Science in Computer Science

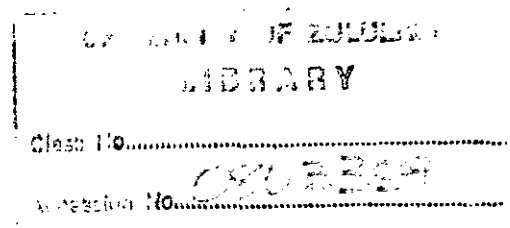
Department of Computer Science, Faculty of Science and Agriculture

University of Zululand

Supervisor: Professor M. O. Adigun


Co-Supervisor: Dr S. S. Xulu

2007



# Declaration

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



---

Signature

# Dedication

To my brother Lawrence

# Acknowledgements

I would like to express my gratitude to my supervisors Prof M.O Adigun and Dr S.S Xulu for their guidance and support to make this work a reality. I would also like to acknowledge the contribution of Dr J. O Emuoyibofarhe and Dr S. O Ojo, who visited my Centre during the course of this work. To my fellow researchers in the centre for mobile e-Services for development, A Ipadiola, K Kabini, J Iyilade, P Mudali, O Olugbara, and S Sibiya I say, thank you very much for the help and support you gave me through out the prototype implementation stage of this work. P Tamba-Tamba, S Kabanda and T. C Nyandeni, thank you for the social support you gave me during the course of this study when things were looking gloomy. Last but not least I would like to express my gratitude to S. T. Chikandiwa for the support she gave me throughout the course of this study.

# Table of Contents

|   |      |
|---|------|
| Declaration.....  | i    |
| Dedication.....   | ii   |
| Acknowledgements.....   | iii  |
| Table of Contents.....  | iv   |
| List of Figures .....   | vii  |
| List of Tables.....   | viii |
| Abstract.....   | ix   |
| CHAPTER ONE .....   | 1    |
| INTRODUCTION.....   | 1    |
| 1.1 Background.....   | 1    |
| 1.2 Statement of the Problem .....                                    | 4    |
| 1.3 Rationale.....  | 5    |
| 1.4 Goal and Objectives .....   | 6    |
| 1.4.1 Research Goal .....   | 6    |
| 1.4.2 Research Objectives .....                                       | 6    |
| 1.5 Methodology .....   | 7    |
| 1.7 Organisation of the Dissertation .....                            | 8    |
| CHAPTER TWO .....   | 10   |
| LITERATURE REVIEW.....  | 10   |
| 2.0 Introduction .....  | 10   |
| 2.1 Context-awareness and Personalisation in m-Services .....         | 12   |
| 2.1.1 Context- awareness .....  | 13   |
| 2.1.2 Personalisation.....  | 14   |
| 2.1.3 Distinction Between Context-awareness and Personalisation ..... | 17   |
| 2.2 User Preferences for m-services .....                             | 20   |
| 2.2.1 User preference models.....                                     | 21   |
| 2.2.2 User Preferences Mining.....                                    | 29   |
| 2.3 User modelling .....  | 32   |
| 2.4 Middleware for Context-Aware Mobile Services .....                | 36   |

|  |   |    |
|--|---|----|
| 2.5                                      | Personalised Context-aware Service Discovery, Selection and Composition ..... | 40 |
| 2.6                                      | Data Mining .....   | 42 |
| 2.6.1                                    | Association Rule Mining .....   | 43 |
| 2.6.2                                    | Classification .....  | 43 |
| 2.6.3                                    | Clustering .....  | 44 |
| CHAPTER THREE .....                      |   | 48 |
| MODEL DEVELOPMENT .....                  |   | 48 |
| 3.0                                      | Introduction .....  | 48 |
| 3.1                                      | User Preference Modelling .....   | 49 |
| 3.1.1                                    | Preferences Representation: Addressing the lack of user expressiveness .....  | 49 |
| 3.1.2                                    | Addressing the Lack of Intuitiveness in the Preference Measures<br>50         |    |
| 3.1.3                                    | Our User Preference Model.....  | 53 |
| 3.2                                      | User Preference Mining .....  | 54 |
| 3.2.1                                    | Mining Attribute Value Preferences .....                                      | 55 |
| 3.2.2                                    | Computing Item Preference Values.....   | 57 |
| 3.2.3                                    | Strict Partial Order Representation of Preferences.....                       | 60 |
| 3.3                                      | Mining Context-based User preferences .....                                   | 62 |
| 3.3.1                                    | Context Model for Mobile Services .....                                       | 63 |
| 3.3.2                                    | Our Context Model and User Preference Mining.....                             | 65 |
| 3.4                                      | User Preference Mining Algorithms.....  | 65 |
| 3.5                                      | User Modelling for m-Services .....   | 72 |
| 3.6                                      | Infrastructure for User Preference Mining and Usage in m-Services .....       | 73 |
| 3.6.1                                    | UPCAMS Design Considerations .....  | 74 |
| 3.6.2                                    | UPCAMS and its Components .....   | 77 |
| 3.6.3                                    | The User Preference Mining Component .....                                    | 83 |
| CHAPTER FOUR .....                       |   | 84 |
| PROTOTYPE DESIGN AND IMPLEMENTATION..... |   | 84 |
| 4.0                                      | Introduction .....  | 84 |
| 4.1                                      | Design of a Prototype.....  | 84 |

|  |   |     |
|--|---|-----|
| 4.1.1  | The Preference Miner in UML .....                     | 85  |
| 4.1.2  | User Session Warehouse Data Model.....                | 88  |
| 4.1.3  | Context-Based User Preference Profiles.....           | 91  |
| 4.1.4  | Context-Based User Preference Repository.....         | 94  |
| 4.2  | Implementation Environment.....                       | 94  |
| 4.3  | Evaluation of the Preference Mining Framework.....    | 96  |
| 4.3.1  | Effectiveness of the User Preference Model .....      | 97  |
| 4.3.2  | Quality of the User Preference Mining Algorithms..... | 103 |
| CHAPTER FIVE .....   |   | 110 |
| CONCLUSIONS AND FUTURE WORK .....  |   | 110 |
| 5.0  | Introduction .....                                    | 110 |
| 5.1  | Conclusions.....                                      | 111 |
| 5.2  | Future Work. ....                                     | 115 |
| BIBLIOGRAPHY.....  |   | 116 |
| APPENDIX A.....  |   | 125 |
| User Preference Profile DTD .....  |   | 125 |
| APPENDIX B.....  |   | 126 |
| Preference Repository DTD.....   |   | 126 |
| APPENDIX C.....  |   | 129 |
| Class Diagram .....  |   | 129 |
| APPENDIX D.....  |   | 130 |
| Formalisation of the Categorical Preferences as Strict Partial orders..... |   | 130 |
| A.D.1  | Preferences Hierarchies.....                          | 130 |
| A.D.2  | Duality of Preferences .....                          | 131 |

# List of Figures

|   |     |
|---|-----|
| Figure 2. 1: Different levels of Context-aware computing (Barkhuus, 2005).....  | 18  |
| Figure 2. 2: Architecture for distributed personalisation proposed by (Kay et al, 2003).....                            | 35  |
| Figure 2. 3: Context Infrastructure proposed by (Riva, 2004) .....  | 39  |
| Figure 3. 1: Item accessibility, probability of selection, and preferences .....  | 51  |
| Figure 3. 2: Context Meta-Model for m-Services, an Adaptation of the Context Meta-Model from (Holland et al, 2004)..... | 63  |
| Figure 3. 3: Context modelling in an m-Services environment .....   | 64  |
| Figure 3. 8: User Model Architecture for Personalisation in m-Services .....  | 73  |
| Figure 3. 9: User Preference Centred Architecture for Mobile Services (UPCAMS).....                                     | 79  |
| Figure 4. 1: Prototype Use Case Diagram.....  | 86  |
| Figure 4. 2: User Preference Miner Activity Diagram. ....   | 89  |
| Figure 4. 3: The ERD for the data warehouse model .....   | 91  |
| Figure 4. 4: User Preference Profile .....  | 92  |
| Figure 4.5: User Preference Repository.....   | 93  |
| Figure 4. 6: The User Preference Miner Test environment .....   | 95  |
| Figure 4. 7: User Preference Miner Performance evaluation environment .....   | 96  |
| Figure 4. 8: Experimental set up for evaluating the preference mining framework .....                                   | 99  |
| Figure 4. 9: Preference Recall and Precision (CWA) .....  | 100 |
| Figure 4. 10: Preference Recall and Precision ( $\neg CWA$ ) .....  | 101 |
| Figure 4. 11: Preference Recall and Precision ( $\neg CWA$ , Combined Data).....  | 102 |
| Figure 4. 12: Scalability of our preference mining with increases in the number of tuples.....                          | 104 |
| Figure 4. 13: Scalability of data access from the data warehouse with increases in the number of Tuples. ....           | 104 |
| Figure 4. 14: Scalability of the matching algorithm with increases in the number of tuples. ....                        | 105 |
| Figure 4. 15: Scalability of the k-Means algorithm with increases in the number of tuples.....                          | 105 |
| Figure 4. 16: Scalability of the DAG with increase in the number of tuples.....   | 106 |
| Figure 4. 17: Scalability of the k-means algorithm with increase the number of clusters.....                            | 107 |
| Figure 4. 18: Scalability of the k-means algorithm with increase in the number of items to be clustered.....            | 107 |
| Figure 4.19: Scalability of the DAG algorithm with increases in the number of items to be represented.....              | 108 |
| Figure 4. 20: Effect of the number of clusters on the total execution time as the number of tuples increases. ....      | 109 |

# List of Tables

Table 2. 1: Middleware aimed at supporting the development of mobile context-aware applications..... 38

Table 2. 2: Comparative analysis of some clustering algorithms ..... 46

Table 3. 1: Notation and Definitions ..... 55

# Abstract

Challenges to the field of Human Computer Interaction (HCI) arising from the emergence of mobile computing can be solved by tailoring the access and use of the mobile services to user preferences. User preferences are traditionally assumed to be static, but due to the dynamic nature of the mobile computing environment, this assumption no longer holds. In an m-Services environment user preferences are not only transient, but they also vary with the changes in context. Furthermore, the assumed preference models do not give an intuitive interpretation of a preference and lack user expressiveness.

To address these issues, this research work defines a user preference model for a context-aware m-services environment, based on an intuitive quantitative preference measure and a strict partial order preference representation. We present some user preference mining algorithms and a framework for context-based user preferences mining in an m-Services environment. The developed user preference modelling and mining framework was prototyped and evaluated it terms of its quality and effectiveness. The user session data for the evaluation of the framework was generated using MATLAB 7.1's Generalised Pareto Probability Density Function (gppdf) with shape, scale and threshold parameters of 1.25, 1, and 0 respectively.

The framework was found to be relatively promising in terms of its effectiveness. The user preference mining framework was also found to relatively scale well with increases in the volumes of data.

# CHAPTER ONE

## INTRODUCTION

### 1.1 Background

The advent of pervasive computing, the proliferation and wide spread adoption of mobile telephony technology, and the information overload on the web (on the contrary), have given birth to a new breed of research in methods, techniques and ideas aimed at providing efficient Mobile Services (m-Services). Because of the mobility of m-Services clients and characteristics of wireless communication, the m-Services operating environment is much more dynamic and has a lot of resource and spatial limitations. In order to retain system quality and usability under these circumstances, research challenges within a number of areas must be solved. Human Computer Interaction (HCI) issues present one of the major challenges in this context. Progress is being made, albeit slowly towards resolving outstanding issues associated with exploiting the mobility of the mobile devices, their personal, "always on", "always with" nature and their ability to make wireless connection for transfer of information. Using the always on, always with nature of the mobile devices and their ability to make wireless connection for transfer of information, the context of the user can be obtained from diverse

information sources. Through the exploitation of their mobile and personal nature, the mobile computing environment can be able to personalise its interaction with the user.

It follows from the foregoing that techniques that have emanated from both the fields of context-aware computing and personalisation can be exploited in addressing challenges to the field of HCI arising from the increasing popularity of mobile computing in today's world.

Over the past decade, researchers have paid significant attention to Context-aware computing. Concerns have been raised that research in context-aware computing risks losing sight of the traditional object in mobile computing, the user, and switch to the context around him (Jameson, 2001). This does not call for a change of focus in context-aware computing but it calls for an expansion of focus in context-aware computing to include user modelling and advanced personalisation techniques. An analytical study of existing scholarship on context-aware computing revealed that there is no clear distinction between context and personalisation in mobile computing. Even though a clear separation of these two concepts seems to be difficult to realise, it has significant benefits to mobile computing application development. These include independent modification of the context network, independent usage of techniques established from context-aware computing and user modelling, development of non personalised context-aware applications.

Advanced personalisation of web applications requires a careful dealing with the user preferences. There are three main areas where user preferences can be used in a mobile computing environment, and these are:

- I. User preferences for device and user interface settings,
- II. User preferences for personalised automatic service discovery, selection, and composition, and
- III. User preferences for personalisation within m-service applications.

Traditionally user preferences are captured in to the system explicitly from the users. However, a closer look at the nature of user preferences in an m-services environment reveals that user preferences do not hold in general (Holland et al, 2004; Gorgogline et al, 2006). They are heavily dependent on the dynamic user, environmental and application context. Thus user preferences change as the user context changes. In this case, it will be a very cumbersome exercise for users to explicitly give their preferences for each and every context. Recent work (Holland et al, 2004; Tseng and Lin, 2005) in the development of personalised web applications is moving towards the use of data mining techniques for extracting user preferences from user session data. Data mining based preference models are collectively known as Automatically Analysing Preference models (Jung et al, 2005). None of the existing research work on Automatically Analysing Preference models has attempted to extract context-based user preferences from the user data.

The mined user preferences are only as accurate as the preference model on ground. Automatically Analysing Preference models lack preference measures that give an intuitive measure and interpretation of a preference (Jung et al, 2005). Current user preference measures use scores or they just distinguish liked and disliked items (Holland et al, 2003). Furthermore, most of the existing preferences models are found wanting when it comes to representing real user preferences because they lack user expressiveness, "I like A more than B" semantics.

## 1.2 Statement of the Problem

The background information given this far amount to the following issues which this research work addressed:

- i. Finding ways of representing/modelling user preferences that gives *an intuitive interpretation of a preference and user expressiveness* to enhance user models for advanced personalisation in the provision of m-Services;
- ii. Mining of context-based user preferences from user session data; and
- iii. Representing user models as an integral part of advanced personalisation in the provision and access of m-services.

In order to address the issues raised above this research:

- established a distinction between context and personalisation in mobile computing;
- developed an automatically analysing user preference model for an m-services environment that has an *intuitive measure* of user preferences and an *intuitive representation* of preferences (user expressiveness);
- developed some algorithms for mining context-based user preferences; and
- developed an infrastructure for user preference mining and usage in a mobile computing environment.

### 1.3 Rationale

The drive towards mobile devices is leading to the integration and convergence of various technologies into a wide range of innovative mobile applications. This, coupled with the affordability and popularity of mobile devices amongst all communities including even the previously technologically disadvantaged communities, makes mobile applications undoubtedly the next wave in the evolution of e-business. Given this background, the envisaged user preference mining framework will foster the development of context-aware and personalised applications, which will make some impact on the evolution of mobile e-business. Improving the quality of interaction of user with mobile computing systems will see users spending less time to access services thereby subsequently reducing the cost of accessing services.

## 1.4 Goal and Objectives

### 1.4.1 Research Goal

The goal of this work was to develop a user preference model and mining algorithms for a mobile computing environment, and to subsequently develop a framework to support the mining and integration of user preferences into the access and provision of mobile services.

### 1.4.2 Research Objectives

The specific objectives of this research work were to:

1. develop a user preference model that gives *user expressiveness* and an *intuitive interpretation* of a preference;
2. develop user preference mining algorithms suitable for mining context-based user preferences in a mobile computing environment based on the model developed in (1);
3. develop a framework for user preference mining that integrates in an m-Services environment; and
4. implement a prototype of the context-based user preference miner for experimental evaluation of framework developed in (3) .

## 1.5 Methodology

The research approach for this study was both theoretical and formulative. To this end, a number of research methodologies were considered in an attempt to realise the objectives of this study.

The theoretical aspect of this research work involved literature review on the work that has been done under context-awareness and personalisation in mobile computing, and user preference modelling and mining. The formulative aspect involved the use of knowledge gained from the literature survey for model formulation, and *proof of concept through mathematically based conceptual analysis and implementation of a prototype.*

Different user preference modelling/mining and user modelling approaches were surveyed. Critical evaluative and comparative analysis of existing processes/algorithms/methods in user preference modelling and user modelling, related to the research challenges of this work were carried out. The above analysis was used to formulate the processes/methods/algorithms that contributed towards achievement of the objectives of this study. An abstraction of the solution to the research challenges raised in this study was given in the form of an architecture, which showed the components involved and how they are glued together.

Time and space complexity of the developed algorithms were analysed to check whether they are computationally feasible.

A prototype of the proposed framework for mining of context-based preferences was implemented and evaluated as a proof of concept. The prototype was then used to evaluate the accuracy of the preference model developed in this work and for performance analysis of the mining algorithms.

## 1.7 Organisation of the Dissertation

This dissertation is organised as follows:

Chapter Two presents background concepts which form the foundation of this work. It covers an analysis of existing scholarship on the issues raised in this research work. It starts by introducing the concepts of context-awareness and personalisation in mobile computing and then digresses into an analysis of the work done on user preference modelling and mining. It also reviews the state of the research into development of context-aware middleware and some classical data mining techniques. Using the bottom-up approach Chapter Three describes the solution to the research challenges raised in this work. It starts by presenting solution to the lack of an intuitive measure of a preference and the lack of user expressiveness in the existing user preference models, and then addresses how the provided solutions can be used to mine context-based user preferences in a mobile computing environment. Chapter Four presents the design,

implementation and evaluation of the user preference mining framework modelled in Chapter Three. Chapter Five summarises the results and briefly discusses how the objectives were met. This chapter concludes the dissertations and overviews the researcher's future research direction.

# CHAPTER TWO

## LITERATURE REVIEW

### 2.0 Introduction

Challenges in computing in relation to the provision of personalised mobile services can be tackled by considering techniques that have emanated from approaches developed through user modelling and context-aware computing. Most of the areas under context-aware computing, such as context capturing, extraction and interpretation, have been covered in many research studies (Biegel and Cahill, 2004; Riva, 2004; Schmidt et al, 1999; Dey and Abowd, 2000). Separate studies have also looked at personalisation (e.g (Kay and Kummerfeld, 2003)). The combination of context-aware computing and personalisation has not been fully explored and the potential benefits have not been fully exploited (Mantjarvi et al, 2001).

The excitement in the research in Context-aware computing has led to fears that the focus on the design of context-aware applications might switch from the traditional subject, *the user*, to the context around him (Jameson, 2001). Developments in context-aware computing indicate that too much attention on context-awareness has overshadowed the importance of personalisation, so

much that there is no longer a clear distinction between context-awareness and personalisation. Our work will attempt to normalise these two extremes by first, *establishing a distinction between context-awareness and personalisation in mobile computing*. Second, by developing a framework for personalisation in the access and provision of m-services, that exploits techniques that have emanated from both the field of context-aware computing and personalisation. The distinction between these two very important concepts of mobile computing will prepare the ground for the development of the *framework for user preference mining and usage in an m-Services environment*.

In this chapter we review the existing scholarship around the issues prevailing in this dissertation. Section 2.1 establishes our working definitions for context and personalisation in mobile computing through the analysis of the work done in classical context-aware computing and personalisation. An analysis of preference models and a review of some preference mining techniques are presented in Section 2.2. In Section 2.3 we discuss the body of literature around user modelling in an m-services environment. Section 2.4 reviews some research efforts towards context-aware middleware to provide tool support for developing context-aware applications. In Section 2.5 we discuss current research efforts towards service registries that support context-awareness and personalisation at middleware level. In Section 2.6 we discuss some classical data mining techniques.

## 2.1 Context-awareness and Personalisation in m-Services

As highlighted earlier context-aware computing has been a very active area of research for the past decade. There is no doubt that context-awareness is very important to mobile computing. While most of the work in this area has concentrated mainly on the development of techniques for abstracting high level context from low level context atoms, another thread of research in this area has concentrated on how context can be used to effectively improve the quality of interaction between the user and the computing environment. This has led to quite a lot of research activity focusing on the development of middleware infrastructures (which will be discussed in Section 2.6) to support adaptation to context both at middleware and application level. However, all these efforts will be void if context is not well defined and all the ambiguities that come with how context is defined are not ironed out.

As highlighted in Chapter 1, the work that has been done to date does not give a clear distinction between context-awareness and personalisation. In this section we will survey the work that has been done so far in context-aware computing and personalisation in relation to mobile computing, with a view to arriving at some working definitions that distinguish these two concepts in mobile computing. An investigation of the state of the art has revealed that little if any

has been done to achieve this goal. However, Coppola et al (2005) highlighted the importance of a clear separation between context and personalisation and the potential benefit thereof. In search of a distinction between context-awareness and personalisation, the following three sub-sections are going to review the existing scholarship on context-awareness and personalisation and then draw out a distinction between these concepts in mobile computing

### 2.1.1 Context- awareness

While most people tacitly understand what context is, they find it difficult to elucidate. This is because context is in itself contextual. Most of the early work on context-aware computing define context by giving an enumeration of examples or by choosing synonyms for context (Dey and Abowd, 2000). This, coupled with the fact that it was not clear whether context in relation to computing should refer to the user or application context, made it extremely difficult to apply context in practice. The lack of standardisation of the techniques which were emanating from the research in context-aware computing further complicates this challenge. Advances in the work on context-aware computing had led to more general definitions of context that include either, both the user and the application context, or the context of the interaction between the user and the computing environment. Out of this development came the most widely accepted definition of context given by Dey and Abowd (2000). They defined context as:

*any information that can be used to characterise the situation of an entity. An entity can be a person, an object or a place that is considered*

*relevant in the interaction between an application and the user including the user and the application themselves.*

The above definition is more general and considers both the application and the user context. In this case the user, his profile, interests, behaviour and preferences become context to the application (or the interaction between the user and the computing environment). According to this definition an application is said to be context-aware if:

*it uses context, as defined above, to provide relevant information and/or services to the user ,where relevance depend on the user's task.*

This definition of context includes personalisation information as context information. This is evidenced by the fact that research efforts based on this definition e.g. (Riva, 2004), take user preferences, interests, user profile as context information. From the point of view of the above definition personalisation is taken as a subset of context-awareness.

## **2.1.2 Personalisation**

Taking heed of the call by Jameson (2001) to expand the focus of context-aware computing to take a more user-oriented approach to building user adaptive systems, and the call by Coppola et al (2005) to separate personalisation concerns from context concerns in mobile computing; in this section we define personalisation and investigate how it can be differentiated from context in mobile computing.

The history of personalisation can be tracked as far back as the early user adaptive interfaces, personal assistants/agents, and adaptive information retrieval (Goker and Myrhaug, 2002). Most approaches originated from desktop computing with user's needs, preferences and expertise. Challenges to the field of HCI in mobile computing bring personalisation to a sharper focus, with much more emphasis on the user's preferences and their dependence on the dynamic user context. Not much is available in literature on personalisation of m-services. One research effort towards this direction is the work by Jostard (2006). Jostard defines personalisation in mobile services as:

*the ability to allow the user U to adapt, or produce, a service A to fit user U's particular needs, and that after such personalisation all subsequent service rendering by service A towards user U is changed accordingly.*

Jostard's definition leaves a lot to be desired for personalisation in the access and provision of m-services. First, the phrase "*all subsequent service rendering by service A towards user U is changed accordingly*" assumes that the user's personalisation information is static across different context descriptions. Second, it states that personalisation is done by the user. This will not be ideal for provisioning of services in a mobile environment due to the dynamic nature of the computing environment and user activities. Ideally, for m-services, personalisation is supposed to be context-based and in principle it is regarded as system driven (Yang et al, 2006). The user should only initiate the process by specifying whether he needs his interaction with the system or services personalised or not.

The major challenge to achieving advanced personalisation of mobile services of the nature discussed above is how to make use of contextual information and exploit the change of context in the personalisation process, which most of the work on personalisation miss out. Attempts to address this problem from context-aware computing rather than from personalisation angle, have led to the current mix-up between context and personalisation in mobile computing. Personalisation in mobile computing should contextually adapt content or services in order to enhance the quality of the user's interaction with the applications.

Some of the recent work in personalisation has been directed towards personalisation of web-based applications. Since most of the applications that run on mobile devices are web based, approaches that have emanated from personalisation of web based applications/services can be taken advantage of to form a basis of personalisation in m-Services.

Based on the arguments presented above, the definition of personalisation in web services given by Bonett (2001) is closer to a practical definition personalisation in a mobile computing environment. Bonett (2001) defines personalisation of web services as:

*the process of gathering user information during interaction with the user, which is then used to deliver appropriate content and services tailor made to the user's needs and preferences.*

One more important idea which this definition provides which most definitions of personalisation leave out is the process of gathering user information. The importance of this is based on the fact that most applications are not designed to support personalisation and hence personalisation data is not readily available. Extra effort is needed to be able to gather such data.

### 2.1.3 Distinction Between Context-awareness and Personalisation

Having discussed different ways researchers look at personalisation and context-awareness, we now present our working definitions for context and personalisation in mobile computing. As highlighted by Coppola et al (2005) and from the foregoing discussion, it is not easy to establish a distinction between context and personalisation in mobile computing. No studies have specifically attempted to establish a distinction between these two concepts in mobile computing. One interesting work that went close to this goal is that of Barkhuus (Barkhuus, 2005). Barkhuus defined three levels of interaction in context-aware computing which are *personalisation*, *passive context-awareness* and *active context awareness* based on the work on context-aware computing he/she reviewed (see Figure 2.1). According to the definition of passive context-awareness (passive context-awareness involves merely presenting the updated context to the user and letting the user specify how the application should change, if at all) given in (Barkhuus, 2005), it can be deduced that passive

context-awareness is irrelevant to the scope of this dissertation. This coupled with the classification done in Figure 2.1 leaves us with two broad mutually exclusive categories of what most researchers refer to as context-awareness in computing; personalisation and active context-awareness.

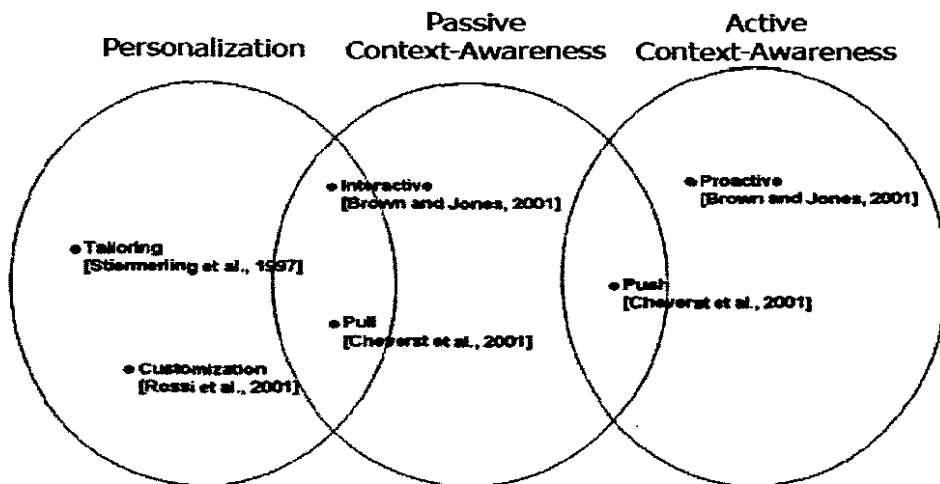


Figure 2. 1: Different levels of Context-aware computing (Barkhuus, 2005)

Although research efforts (e.g. Toivonen, 2004, Coppola et al, 2005)) in the development of context-aware middleware for mobile applications are already going towards the separation of user profiles from context, they are silent as to how this separation is to be achieved. Our approach to this problem is to consider these concepts from the angle of developing non-personalised context-aware services/applications. From this point of view and the definition of context given by (Dey and Abowd, 2000), we define context as follows:

*Context is any information that can be used to characterise the situation of an entity that is not specific to the user or a user group. An entity is a person, place, or object that is considered important to the interaction between a user and an application, including the user and the application themselves.*

This definition implies that when the interaction of the user and the computing environment is subjected to a certain context the system must adapt itself in the same way for different users. If it adapts differently it therefore means that the environment has been subjected to some personalisation some how.

This research work defines personalisation information as:

*any information that can be used to tailor/adapt the interaction of a user with a system or service to the needs and preferences of the user or user group.*

From the definition of personalisation given by Bonett (2001) we now define personalisation in m-services as:

*the process of gathering personalisation information and the subsequent use of the gathered information to tailor/adapt the interaction of a user with a system or service to the needs and preferences of a specific user or user group.*

According to this definition, personalisation refers to all processes that involve the gathering of data specific to a given user or user group and the subsequent tailoring of the interaction (based on this data or on the inferences drawn from this data) between the user and the computing environment to the user. The user specific ( or user group specific) data and the inferences drawn from the data are what we refer to as personalisation information. On the other hand, context is what ever information that is not specific to a user or group of users that can be used to characterise the situation of an entity involved in the interaction between the user and a computing system. Our previous work in (Jembere et al, 2006) showed how the distinction between context-awareness and personalisation can

be used to develop a platform for user preference mining and usage in an m-services environment.

## 2.2 User Preferences for m-services

In classical context-aware computing (where personalisation is taken as part of context-awareness), user preferences are taken as part of the user's static context, whilst in reality, some user preferences are not only transient but vary with changes in context. This is drawn to a sharper focus in a mobile computing environment due to the dynamic nature of user, application and environmental context. In this case, user preferences and context need to be modelled as separate entities at different levels, to enable exploitation of context data in the personalisation process. This is another point, where the above discussed separation of context-awareness and personalisation in mobile computing is very important.

Traditionally, user preferences are captured as user input into the system through the query interface and user feedback mechanism. However, given the dynamic nature of context in mobile computing and the dependence of user preferences on context, it will be a very cumbersome task for the user to explicitly give his/her preferences for each and every context description. This research work envisages mining of context-based user preferences from the user session data as one of the solutions that can be used to address this challenge. The user session data will be extracted from the interaction between the user and the

computing environment, both at service selection level and within services. The remainder of this section is organised as follows: In Section 2.3.1 and Section 2.3.2, we discuss issues around user preference modelling and mining respectively.

## 2.2.1 User preference models

Current personalisation techniques are inhibited by use of preference models with limited expressiveness (Holland et al, 2003). Most of the traditional user preference modelling approaches either use scores to describe preferences or just enumerate liked and disliked values. Thus, in these models, real preferences in the form “I like A more than B” can not be expressed in a natural way. Such user preference models are said to be lacking user expressiveness. Further to the lack of user expressiveness, these models also lack preference measures that give an intuitive interpretation of a preference. Automatically analysing user preference models, depending on how they represent preferences, can be classified into the following three classes: Vector similarity, Probability, and Association rule based preference models (Jung et al, 2005).

### **A. Vector Similarity based Preference Models**

Vector similarity-based preference models use similarity between users and/or items to predict the active user’s (the user for whom preference predictions are to be made) preferences. Vector similarity based preference models can be broadly

categorised into two groups and these are: *Collaborative Filtering* and *Content-Based Filtering*. Collaborative filtering refers to a set of methods that uses user prior preferences to predict new ones based on the preference similarities among users. A typical collaborative filtering algorithm has the following two stages:

1. *Finds the similarity between the active user and the users in the database.*
2. *Use the similarities found in one to predict the preferences of the active user.*

In content-based filtering each active user is assumed to operate independently. The user preference on an item is determined by the similarity between the contents/attributes of the items the user has selected before and the target items. As can be deduced from the definitions above, when preferences are represented as vector similarities, it is hard to get the intuitive interpretation about how much a user dislikes or likes an item and which items are preferred more than others.

## **B. Probability-based Preference Models**

Probability-based preference models determine the probability that a user selects an item from the historical user data and use it predict which items the users are likely to select. The problem with probability based approaches is that when preferences are represented as probability that a user selects an item, a preferred item with low frequency of selection or not so preferred items with high frequency of selection, is not measured correctly.

### **C. Association Rule-Based Preference models**

These preference models generate user preferences through scanning the database for some association rules that can be used to generate user preferences. These association rules can be between users and or between items. The concept of an association is different from that of a preference in that association deals with relations (relation in terms of occurring together) between item sets or relations between users rather than the intuitive interpretation of a given user's preference on an item. Apart from the lack of intuitiveness in the preference measure in association rule based preference models, the rules discovered can be spurious and irrelevant (Adomavicius and Tuzhlin, 2002) and one of the major problem which arises is how can these rules be validated without human intervention.

Against this background, researchers from both mathematics and computer science (e.g. Jung et al, 2005; Kiessling, 2002; Kiewera, 2005) have been working towards developing intuitive preference models. Our research work takes inspiration for user preference modelling from two frameworks: the Strict Partial Order preferences framework (Kiessling, 2002) and the feature preferences framework (Jung et al, 2005), which will be discussed in the following two subsections.

### 2.2.1.1 The Strict Partial Order (SPO) Preferences Model

Kiessling (2002) introduced a very expressive and mathematically well-founded framework for preferences. This framework is based on strict partial order relationships. A **strict partial order** is a binary relation  $R$  that is irreflexive, transitive, and asymmetric. In other words, for all  $a$ ,  $b$ , and  $c$  in a given set  $P$ , we have that:

- $\neg(aRa)$  (irreflexivity):  $a$  can not be preferred to itself
- if  $a \neq b$  and  $aRb$  then  $\neg(bRa)$  (asymmetry): if  $a$  is not the same item as  $b$ , and  $a$  is preferred to  $b$ , then  $b$  can not be preferred to  $a$
- if  $aRb$  and  $bRc$  then  $aRc$  (transitivity): If  $a$  is preferred to  $b$  and  $b$  is preferred to  $c$ , then  $a$  is preferred to  $c$

From this definition of a strict partial order relationship, it is clear that strict partial order relation offer a natural way of representing preferences.

The work in (Kiessling, 2002) defines a preference  $P$  as a strict partial order  $P = (A, \prec_P)$ , where  $A = (A_1, A_2, \dots, A_k)$  denotes a set of attributes with corresponding domains  $(A_i)$ . The domain of  $A$  is defined as the Cartesian product of the  $dom(A_i)$ ,  $\prec_P \subseteq dom(A) \times dom(A)$  and  $x \prec_P y$  is interpreted " $y$  is better than  $x$ ".

A set of intuitive preference constructors for base and complex preferences is defined. The constructors for base preferences on categorical domains are  $POS(A, POS\text{-set})$ ,  $NEG(A, NEG\text{-set})$ ,  $POS/NEG(A, POS\text{-set}; NEG\text{-set})$ ,

$POS/POS(A, POS1\text{-}set, POS2\text{-}set)$  and  $EXP(A, E\text{-}graph)$ . The  $POS\text{-}set \subseteq dom(A)$  of a POS preference defines a set of items that are preferred to all the other items of  $dom(A)$ . Analogously, the  $NEG\text{-}set$  of a NEG preference defines a set of items that are less preferred than all the other items in  $dom(A)$ , i.e. any other items which are not in the  $NEG\text{-}set$  are preferred to any of the items in the  $NEG\text{-}set$ . The POS/NEG preference is a combination of the previous preferences, where items in the  $POS\text{-}set$  are preferred to all the other items, items in the  $NEG\text{-}set$  are the least preferred and any other items neither in the  $POS\text{-}set$  nor the  $NEG\text{-}set$  are preferred to the items in the  $NEG\text{-}set$  and less preferred to the items in the  $POS\text{-}set$ . In POS/POS preference an optimal set of items ( $POS1\text{-}set$ ) and an alternative set ( $POS2\text{-}set$ ) can be specified. The items in  $POS1\text{-}set$  are the most preferred, followed by the items in the  $POS2\text{-}set$ . All the items not in the  $POS1\text{-}set$  and  $POS2\text{-}set$  are less preferred. In Explicit-graph ( $E\text{-}graph$ ) of an EXPLICIT preference a user can specify any better-than relationships. An  $E\text{-}graph$  is a directed “better than” acyclic graph. All items in an acyclic graph are better than all other items in  $dom(A)$ .

The preference constructors for numerical domains include  $AROUND(A, z)$ ,  $BETWEEN(A, [low, up])$ ,  $LOWEST(A)$  and  $HIGHEST(A)$ . In an  $AROUND$  preference the desired value is  $z$ , but if it is not available, then values with nearest distance apart from  $z$  are best alternatives. For a  $BETWEEN$  preference the values within  $[low, up]$  are optimal. For  $LOWEST$  ( $HIGHEST$ ) preferences lower (higher) values are better.

Preferences can inductively be combined with complex preference constructors. A Pareto preference  $P = P1 \otimes P2$  treats the underlying preferences as equally important and a Prioritized preference  $P = P1 \& P2$  treats  $P1$  as more important than  $P2$ .

The strict Partial Order user preferences model was designed for an environment where the user explicitly specifies his/her feature/attribute preferences and the order in which there are to be effected. Hence the complex preferences constructors to combine preferences on attributes to get the preference (Pareto and Prioritised preferences) on an item were defined accordingly. This makes it a qualitative user preference model. This then poses a problem in applying the framework with automatically analysing preference models. Automatically analysing preference models require a quantitative preference model. Furthermore, it will be difficult to use the models without human intervention.

### 2.2.1.2 The Feature Preference Model.

Quantitative preference models lack preference measures that give an intuitive measure of a preference (Jung et al, 2003; Jung et al, 2005). Most of them use the frequency of selection of an item (directly or indirectly) as a measure of a preference on an item. Frequency of selection fails in this respect based on the following arguments:

- Although a user likes an item, the selection frequency can be low if the item is rarely distributed in that domain.
- On the other hand, although a user may not like an item, its selection frequency can be high if the item appears very frequently in that domain.

Thus, to get an intuitive measure of a preference on a set of items, the accessibility of the items in question should be considered. Jung et al (2005) presented a very intuitive measure of a preference based on the fact that the probability that a user selects an item is determined by both the preference and the accessibility of the item.

Intuitively, a user's preference for an item is determined by the user's preferences for the features of the item. This is generally assumed in most preference models, including the Strict Partial Order preference model discussed above and in most content-based vector similarity preference models. An intuitive preference model, which considers this fact and the accessibility of an item in determining the user's preferences on an item, was presented in Jung et al (2003, 2005). Their work defines a *preference* as *the concept where a relationship is made between a person and a target item which contains several kinds of attributes/features*. From this point of view, preference for an item can be expressed as a function of preferences for the attributes contained in the item. A preference was then mathematically represented as a function of item  $x$  and the user profile  $G$ , where the user profile will be approximated by the User History  $U$ , i.e

$$Pref(X) = f(x, G) \approx f(x, U) \quad (2.1)$$

The user history is represented by a set of selected items  $U = (x_1, x_2, x_3, x_4, \dots, x_n)$ .

Each item has a set of several attributes/features denoted by  $w$ . Item  $x$  is then defined as a set of features,  $x = (w_1, w_2, w_3, \dots, w_m)$ . The User Profile  $G$  is defined

by the preference of each feature  $w$ , i.e.  $G = \{pref(w_1), pref(w_2), pref(w_3), \dots\}$ .

The feature preference  $Pref(w)$  is computed from the user history. Since the user profile  $G$  and the item  $x$  can be represented by common attributes,  $w_i$ 's, they can be compared through these attributes. As a consequence, a preference of an item  $x$  is then represented as follows:

$$Pref(x) = \frac{1}{M(x)} \sum_{w \in x} Pref(w_i) \quad (2.2)$$

where  $M(x)$  is a normalisation term which is defined as the number of features in item  $x$ . Mutual information is used as a measure of a feature preference,

$$Pref(w) = I(X(w); U) = \ln \left( \frac{P(X(w); U)}{P(X(w))} \right) \quad (2.3)$$

where  $X(w) = \{x \mid w \in x\}$ ;  $P(X(w); U)$  is the feature selection probability given the user history and  $P(X(w))$  is the unconditional feature probability, which gives a measure of the feature's accessibility.

Having discussed how preferences can be modelled, the following section discusses some research efforts towards mining of user preferences from user session data.

## 2.2.2 User Preferences Mining

Over the years, data mining have been typically used to extract information that users can view and use for decision making. Recently, more and more work is now looking at ways of addressing the need for mining information that computer systems will use. The web, its resources and users offer a wealth of information for data mining and knowledge discovery (Yang et al 2004). The availability of resources for data mining on the web can be attributed to the success of applications of data mining techniques in recommender systems. Over the past decade, data mining based personalisation techniques, which include collaborative filtering (e.g. Resnick et al, 1994), content-based filtering (e.g. Paulson et al, 2003), association rule mining (e.g. Shyu et al, 2005; Kim and Kim, 2003), pattern discovery (e.g. Tseng and Lin, 2005), etc, for recommender systems have been developed. The major draw back of these techniques in relation to this work is that none of these techniques explicitly mine for user preferences.

Our work is concerned with mining of user preferences from archived user session data and using them for personalisation of the access and provisioning of m-services. In mobile computing, there are three main levels where user preferences can be used, and these are:

- I. User preferences for device and user interface settings,
- II. User preferences for personalised automatic service discovery, selection, and composition, and

### III. User preferences for personalisation within m-service applications.

The first level does not have enough resources to support mining of user preference mining and hence preference elicitation is restricted to user input. The availability of data in the form of web logs in the last two levels makes them lucrative areas for application of data mining techniques as a way of soliciting user preferences. The next two subsections discuss some research efforts towards mining of user preferences at both these two levels.

#### 2.2.2.1 Mining User Preferences for Automatic Service Discovery and Selection.

To the best of our knowledge no studies have explored the area of mining user session data that contain both the context histories and service request data, in order to get actionable knowledge on user preferences for context-based service discovery, selection, and composition. An effort closer to this goal is the work by Tseng and Lin (2005) for location based services. The research conducted by Tseng and Lin (2005) did not involve user preference mining per se, but it proposed a data mining method, SMAP-Mine, that can discover patterns of sequential movement associated with requested services for mobile users in a mobile web system. These patterns will then be used for wireless applications like data allocation, data replication; location based personal agent and context-aware and personalised services.

## 2.2.2.2 Mining User Preferences for Personalisation within

### Applications

Most of the research efforts in user preference mining have been directed towards mining of user preferences for personalisation within applications. Most approaches to preference-based recommender systems such as collaborative filtering; context-based filtering and their variants have over the years found wide applications in e-Commerce applications (Scafer et al, 2000), personalised news recommender systems (Tintarev and Masthoff 2006; Wang et al, 2004), Digital TV program recommendation system (Jung et al, 2003). As discussed earlier, these approaches lack intuitive preference measures and representations, hence in this section we only focus on the approaches that are closely related to this study.

As a follow up to the framework developed in (Kiessling, 2002) the work in (Holland et al, 2003) presents an approach for automatically mining strict partial order preferences from e-commerce user log data. The user preference mining framework to be developed in this study builds upon the work done by Holland et al (2003) with the aim of filling the following gaps:

- The approach uses frequency of selection as a measure of a preference. As discussed earlier this does not give an intuitive measure of a preference, and

- In a mobile computing environment, user preferences depend on the dynamic user, application and environmental context. The approach does not provide a mechanism of mining context-based user preferences.

## 2.3 User modelling

One fundamental prerequisite for applications to be able to adapt to the user's needs and preferences is that the system requires a representation of the user's preferences, interests, behaviour, and any other user specific information. Such a representation is typically conveyed as a user model and hence why personalisation in computing is sometimes defined as the adaptation to user models (Bonett, 2001). User modelling is an important part of mobile computing. It is essential for the personalisation of the user environments and the services/information that a mobile user accesses. Considering the enormous growth of web content and web based applications and the importance of customisation of service access in a mobile computing environment, there is a need for the development of user models that can be reused and shared among different types of applications. This paradigm is known as cross personalisation. The emergences of technologies such as XML that allow parsing and sharing of information over the web makes cross personalisation a reality. This section discusses some of the existing user modelling approaches with the aim of finding how they can be adapted to support distributed personalisation/cross personalisation in the access and use of m-services.

User modelling approaches exploit the fact that the user revisits the same system with similar requirements (Niederee et al, 2004). Traditional user modelling approaches started with user modelling shell systems which came out of academic research, e.g. UMT, BGP-MS, um, etc (Kobsa, 2001). The increase in the value of personalisation due to the popularity of electronic commerce in the late 1990s led to the development of commercial user modelling servers. The major stride which was taken to move from academic user models to commercial user modelling servers was the decoupling of the user modelling and the applications themselves. In the commercial user models, the user models are not functionally integrated into the application, but communicate with the application through inter-process communication and can serve more than one client application at the same time (Kobsa, 2001).

Apart from the many advantages of centralised user modelling system design, the key advantages which are of particular importance are that the user modelling servers relieve clients from user modelling task and can take advantage of powerful hardware resources. The major drawbacks in the use of commercial user modelling servers are the necessity to connect to the server and potential single point of failure. Due to the network instability in mobile computing, the user models in this environment should not be completely insulated from the application. Furthermore, the need to support mobile ad hoc

networks calls against the complete insulation of user modelling from the application.

Not much is available in literature in the area of user modelling for mobile computing. This research adopts the models developed for ubiquitous computing as a point of origin. Generally, user models in an ubiquitous computing environment have the following characteristics (Byun and Cheverst, 2001):

- Data acquisition for user models is largely from interaction with the user;
- Representation of data in user models can work effectively as a hybrid model of a data and a behavioural model;
- User models need sufficient time to interact with the user for the model to learn the user's behaviour, and
- In general user models do not work efficiently if completely insulated from applications.

Based on the characteristics of user models in ubiquitous computing mentioned above, the architecture shown in Figure 2.2 was proposed by Kay and Kummerfeld (2003). The architecture addresses the issues of distributed personalisation with particular attention to reusability of the user models. It also ensures that the user models are not completely insulated from the applications. The architecture introduces a model, which have a definitive user model, **U**, which can be shared by different applications. Apart from the definitive user

model there are several partial user models. Each application, **A**, has its own associated user partial model **u**.

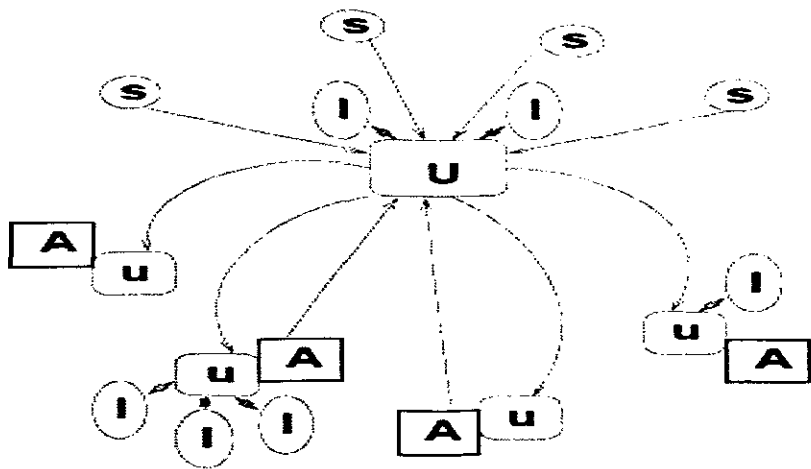


Figure 2. 2: Architecture for distributed personalisation proposed by (Kay et al, 2003).

The inference sources labelled **I** in the Figure 2.2 are internal reasoning mechanisms for each user model and each of these is a potential source of information. The figure also shows some sensors, **S**, which also contribute user modelling information in the form of context data to the user model. Instead of the user model getting context information directly from sensors, the user model can get abstracted context information from the uniform service infrastructure like the one proposed in (Riva, 2004). This will greatly reduce the task of personalised context-aware application developers, since they need not to develop some contexts abstraction components for the user models.

## 2.4 Middleware for Context-Aware Mobile Services

Although, over the past decade, a lot of effort has been directed at finding methods for capturing, representing and using context information, context-aware applications are still not common. Context gathering methods are tightly integrated into single applications and cannot be reused or shared among applications. Each new application must be built from the ground up, thus requiring a large effort and nontrivial knowledge. This results in the development of complex applications and it makes it difficult to transfer existing applications to new environments with different sensing technologies. The most widely proposed solution to this challenge (e.g. Dey and Abowd, 2000; Altvin, 2003; Fahy and Clarke, 2004; Riva and flora, 2006) is decoupling application development and context abstraction methods.

Several approaches have been proposed to decouple context abstraction and application development. The first one is the *tool kit approach* developed by Dey and Abowd (2000). The tool kit approach provides a number of reusable components to support rapid prototyping of context aware applications. Besides the tool kit approach, *middleware infrastructures* have been proposed. Over the past few years a lot of research efforts have been directed towards developing context-aware middleware infrastructures for mobile computing (e.g. Altvin, 2003; Fahy and Clarke, 2004; Riva and Flora, 2006; Gu et al, 2004; Gehlen and Mavromatis, 2003). Middleware infrastructures encompass uniform abstractions of reliable services for common operations and support for handling and dealing

with context thus simplifying the development of context-aware applications. An analysis of the research that has been conducted on this subject matter categorised context-aware middleware into two main groups along the line of the purpose for which they have been developed. These categories are:

- Middleware infrastructures aimed at easing the development of context aware applications, and
- Middleware infrastructures aimed at providing adaptive and reconfigurable execution environment.

This research work is more inclined towards easing the development of context-aware applications, but with more emphasis on context-based personalisation. Lesser effort has been put into developing middleware infrastructures that specifically addresses personalisation issues. This is due to the fact that the research assumes personalisation as a subset of context-awareness. Table 2.1 shows some of the recent research efforts towards development of middleware infrastructures to support development of mobile context-aware applications.

Since our work is based on component-based Service Oriented Architecture, it takes inspiration for designing the middleware for context-aware applications from the work done by (Riva, 2004). Each context provider is a service provider that can be configured by the middleware at runtime according to the client's (application's) requirements. The infrastructure

Table 2. 1: Middleware aimed at supporting the development of mobile context-aware applications

| Middleware   | Design Goal   | Architectural styles                          | Communication paradigm                 |
|--|---|---|--|
| Controy (Riva and Flora, 2006)   | Supporting Multiple Context Provisioning Strategies.  | Client-Server, Peer-to peer                   | Push and pull                          |
| CASS (Fahy and clarke, 2004)   | server-based middleware to counteract the memory computational power limitations in mobile devices  | Client-server                                 | No communication paradigm discussed    |
| CAMUS (Ngo et al, 2005)  | Design consideration for a unified sensing framework, formal modelling and representation of the real world, Pluggable reasoning engines for high level context data, and context delivery runtime service composition mechanism. | Service Oriented Architecture                 | No communication paradigm discussed    |
| SOCAM (Gu et al, 2004 )  | Providing middleware support for the building and rapid prototyping of context-aware mobile services.   | Service Oriented Architecture,                | Publish-subscribe model                |
| Conceptual model for structuring Context-aware applications (Riva, 2004) | Decoupling of applications and the actual process of context extraction process using a uniform service infrastructure  | Component based Service oriented Architecture | - Publish/subscribe<br>- push and pull |
| TOTA (Mamei, 2003)   | Supporting uncoupled and adaptive interaction between application components.   | Peer- to-peer                                 | Tuple space communication model        |

proposed by Riva (2004) is designed in such a way that it provides a uniform abstraction of context, facilitates easy creation of context-aware applications, reusing software and hardware designs. It also provides easy system maintainability and rapid extensibility, and supports collaboration and context sharing among diverse applications. From the point of view of the application designer, such context provider components absolve to the role of common software libraries. From the point of view of the service infrastructure they constitute reusable building blocks for context provisioning that can be added or

ignored according to the application's needs. The architecture is as shown in Figure 2.3.

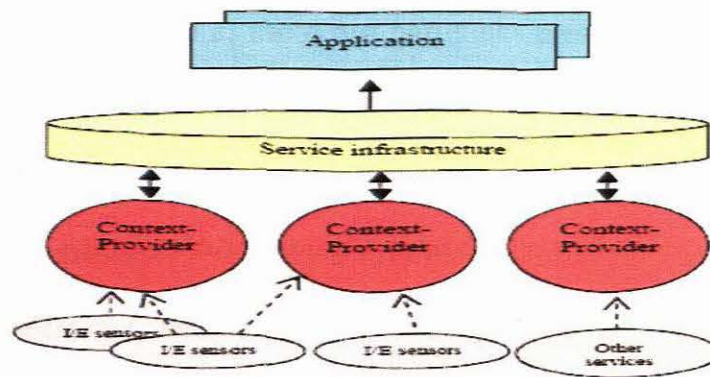


Figure 2. 3: Context Infrastructure proposed by (Riva, 2004)

We envisage an architecture that distributes personalisation among both the middleware and application layers, providing a service oriented point of view, that allows reuse and sharing of user modelling information, software and hardware components. This will not be a completely new approach since most of the current middleware infrastructures consider the user profile as context data, and hence they assume that there are components that provide the user profile. In response to the call by Jameson (2001) to expand the focus on context-aware computing to take a more user oriented approach to building user adaptive systems, this research will divide the uniform service infrastructure proposed by Riva into two modules: namely the Context and the Personalisation module. Our division of context and personalisation will be based on the discussion given in Section 2.1.

## 2.5 Personalised Context-aware Service Discovery, Selection and Composition

As discussed in Section 2.3.2, there are two main areas where user preference mining can be applied in an m-services environment: *within applications* and *at middleware level for mining user preference for automatic service discovery, selection and composition*. Preference models based on the item's features can easily be adapted for mining user preference within applications (e.g. e-commerce applications). Unlike for user preference mining within applications, preferences models based on the item features can not be directly adapted at middleware level for mining preferences for automatic service discovery, selection and composition. This is because of the way the current standardisation in service directories and registries are designed. The user preference model envisaged in this work requires an enriched semantic service description, which can capture the attributes/features of the services. The commonly used, service discovery and selection technologies, UDDI, ebXML and WSDL still essentially lack strong concepts for semantically rich service description (Balke and Wagner, 2003). Approaches from the Semantic Web like DAML-S or WSMF are first efforts that try to provide semantically rich service descriptions through the use of ontology. But even letting aside the problems of different ontology or missing concepts of interoperability the human interaction with services poses severe problems due to the lack of personalisation (Balke and Wagner, 2003). This is mainly because these technologies were designed for business-to-business

interaction. This calls for a paradigm shift in service description technologies to descriptions that support consumer-oriented automation of service discovery, selection and composition. As can be established from the user preference modelling frameworks discussed in Section 2.3, such technologies should represent services in terms of their attributes/features. Efforts towards this goal include the now defunct HP's E-speak (E-Speak, 2001). E-speak describe services as a set of attributes within several different vocabularies which are sets of attributes common to a logical group of services. Lookup requests are matched against service descriptions with respect to these attributes.

Another weakness with the current standardisation efforts regarding service directories is that they are not designed to support mobility and context-aware service discovery. However with the emergence of some research efforts, which include (Doulkeridis et al, 2006; Klan, 2006; Bormann et al, 2005) towards this goal, the future of context-aware service directories looks very bright.

Another interesting work towards this goal specifically aimed at providing a service description technology for pervasive computing environments is Pervasive Service Description Language (PSDL) and its counter part the Pervasive Service Query Language (PSQL) (Thompson and Midkiff, 2005). PSDL does not define the specific content for service descriptions but it generically contains the following information: Device information (make, model, software, resources); Service information (type, attributes); Physical information (location,

accessible items); Methods provided (e.g. WSDL). The PSQL is responsible for describing a service from the client side point of view. It compares the service's attributes to the ones requested by the user and uses that comparison to discover, select and compose services.

However, most of these promising emerging technologies are still very immature and lack standardisation, which is a necessity in this area. The design of service registries and directories that provide a semantically rich way of describing services that will cater for personalisation, contextualisation and mobility issues in service discovery, selection and composition is still an open area which needs further research.

## 2.6 Data Mining

Data mining is a multi-disciplinary field with threads from Statistics, machine learning, pattern recognition and Artificial Intelligence. Data mining is defined as the non-trivial extraction of implicit, previously unknown, and potentially useful information from data (Klevecz, 1999). Over the years a lot of data mining techniques have been developed, some of which are discussed in the following sections. This section will give a brief overview of some few data mining techniques which are: association rule mining, classification and clustering. Much more emphasis will be put on clustering, which is the only classical data mining technique used in this study.

## 2.6.1 Association Rule Mining

Association rule mining is one of the most commonly applied data mining techniques for local pattern discovery in unsupervised learning systems (Kantardzic, 2003). Given a collection of sets of items, association rules describe how likely various combinations of items are to occur together in the same set. As discussed earlier association rule mining, though used in some indirect preference models, can not be used to automatically mine for intuitive user preference.

## 2.6.2 Classification

Classification is a process of assigning unlabeled samples to discrete labelled classes. This is achieved by training a function that maps a sample into one of the several predefined classes. Data mining approaches to classification can either be statistical or logical. Statistical classification models are based on learning a classification rule that minimises the Expected Cost of Misclassification (ECM). The ECM provides the cost undergone (in terms uncertainty) in allocating samples in to the predefined classes. Fisher's discriminant function and the quadratic classification rule are examples of statistical classification rules (Kantardzic, 2003). Logical classification models are based on expressions that are evaluated true or false by applying Boolean comparative operators to new samples. Decision trees and decision rules are

typical examples of logical classification techniques. A discussion of the specific classification rules is beyond the scope of this work. To the best of the researcher's knowledge, there are no preference measures which are based on this technique.

### 2.6.3 Clustering

Cluster analysis is a set of methodologies for automatic classification of samples into a number of groups using a measure of association, so that the samples in one group are most similar and samples belonging to different groups are least similar (Kantardzic, 2003). Clustering is one of the most widely used data mining technique for identifying interesting distributions and patterns in the underlying data in large and growing databases.

The fundamental clustering problem is to partition a given set of data in to groups (clusters), where the samples in each cluster are most similar. The main requirements of clustering algorithms are:

- Scalability with data size;
- Dealing with different types of attributes;
- Discovering clusters with arbitrary shapes;
- Minimal requirements of domain knowledge to determine input parameters;
- Ability to deal with noise and outliers;

- Insensitivity to order of input records, and
- High dimensionality Interoperability and usability.

There are quite a number of problems with clustering algorithms. Some of them are :

- Current clustering techniques do not address all the requirements adequately (and concurrently);
- Dealing with large number of dimensions and large number of data items can be problematic because of time and space complexity;
- The effectiveness of the method depends on the definition of “distance” (for distance-based clustering);
- If an obvious distance measure doesn’t exist one must define it, which is not always easy especially in multidimensional spaces, and
- The result of the clustering algorithm (that in many cases can be arbitrary itself) can be interpreted in different ways.

Clustering algorithms can be classified based on how they perform their clustering process, which gives the following categories:

- Hierarchical clustering;
- Partitional clustering, and
- Incremental clustering.

The type of clustering algorithm to use depends on the type of data to be analysed and how the results are to be used. The Table 2.2 shows the weaknesses and strength of each category of clustering algorithms.

Table 2. 2: Comparative analysis of some clustering algorithms

| <b>Class</b> | <b>Strength</b>  | <b>Weakness</b>  |
|--------------|--|--|
| Hierarchical | <ol style="list-style-type: none"> <li>1.No need to assume any particular number of clusters. Any desired number of clusters can be obtained by 'cutting' the dendrogram at the proper level.</li> <li>2.They may correspond to meaningful taxonomies. E.g. taxonomies in biology</li> </ol>                 | <ol style="list-style-type: none"> <li>1. Fail to handle convex and elongated shapes of clusters.</li> <li>2. Construction of dendograms for large data sets is computationally very expensive</li> </ol>  |
| Partitional  | <ol style="list-style-type: none"> <li>1.Time and space complexity relatively low</li> <li>2.Perform better than hierarchical algorithms in application with large datasets</li> </ol>   | <ol style="list-style-type: none"> <li>1.lack of guidelines available for choosing k-number of clusters.</li> <li>2.May result in the break up of genuine clusters</li> <li>3.Ambiguity about the best direction of initial partition, updating the partition, adjusting the number of clusters and stopping criterion.</li> <li>4.Face difficulties in determining clusters with large variances</li> </ol> |
| Incremental  | <ol style="list-style-type: none"> <li>1. Incremental algorithms non-iterative and therefore time requirements are less.</li> <li>2. less space requirements</li> <li>3. Can work with very large data sets with some of the data stored in secondary memory</li> <li>4. Can cluster data streams</li> </ol> | <ol style="list-style-type: none"> <li>1. the generated clusters depends on the order of processing of the samples.</li> <li>2. it starts to regenerate with small increases in database sizes</li> </ol>  |

Clustering is the most widely used data mining technique in vector similarity based preference models, which include collaborative filtering and content-based models. The advantage of clustering techniques over classification techniques towards this goal, is that classification requires that groupings be predetermined

using domain knowledge were as clustering lets the data determine the groups based on the similarities of the items in question. Thus clustering can be used to cluster all the items in a given domain and an inference that items in the same cluster with the items the user have shown interest in are equally preferred, can be made.

This chapter presented some of the work that has been done in trying to solve the research challenges raised in this work and issues that lay the foundation for the solution to these challenges. We started by presenting issues around personalisation and context-awareness in relation to mobile computing. We then presented some user preference modelling and mining techniques that are relevant to this study. We discussed research issues around creating a platform to support user preference mining in mobile computing. Lastly, we have presented some data mining techniques that can be applied in mining user preferences.

# CHAPTER THREE

## MODEL DEVELOPMENT

### 3.0 Introduction

This Chapter presents the proposed preference model and a framework for user preference mining and usage in an m-services environment. Our first research challenge is to find ways of representing/modelling user preferences that gives *an intuitive interpretation of a preference and user expressiveness* to enhance user models for advanced personalisation in the provision of m-Services. Therefore, Section 3.1 addresses these two weaknesses of traditional automatically analysing user preference models. In Section 3.2 we present our proposed user preference mining framework. Our second research challenge is the mining of context-based user preferences from user session data. Therefore, Section 3.3 presents our context model and discusses how our preference mining framework can be adapted for mining context based user preferences in mobile computing environment. Furthermore, in Section 3.4 we present our context-based preference mining algorithms. Our last research challenge is to find a representation of user models in support of advanced personalisation in the provision and access of m-services. To address this challenge, in Section 3.5 we present our approach to user modelling in an m-Services environment. In

Section 3.6 we give an architectural framework to support the mining and usage of context-based user preferences in an m-Services environment.

## 3.1 User Preference Modelling

In this section we address two very important aspects of preference models, which have been identified to be the main weaknesses of traditional automatically analysing user preference models. Traditional automatically analysing user preference models:

1. lack user expressiveness, and
2. employ preference measures that do not give an intuitive measure of a preference.

In Section 3.1.1 we address the lack of user expressiveness and in Section 3.1.2 we deal with the second weakness, namely the lack of intuitiveness in the preference measures. Section 3.1.3 presents our user preference model, based on the solution to the afore-mentioned challenges.

### 3.1.1 Preferences Representation: Addressing the lack of user expressiveness

Preferences are comparative in nature and are considered on a set of given options. It should be possible to identify preferences even amongst disliked items. Current user preference models do not capture this and they are thus said

to lack user expressiveness. In a real life situation, a user is able to make preference relationships even amongst a set of disliked items. That is, if a set of items available contains three disliked items *a*, *b* and *c*, a user is still able to make a preference relationship on these items (e.g. I prefer *b* to *a* and *c*) even though he/she does not like any of them.

So the challenge is to be able to represent these real life cases in a model. This is of great importance in an m-service environment, where a user may be in need of a service to accomplish a particular task. All the services that may be available might not be in the set of services that the user likes but, because the user needs a service to accomplish a task, a service that is most preferred amongst all the available services has to be recommended. To address the problem of user expressiveness, the strict partial order preference model is adopted. In this research work, we therefore define a user preference as follows:

*A **preference** is strict partial order relation on a set of given items.*

### 3.1.2 Addressing the Lack of Intuitiveness in the Preference Measures

A user prefers one item to the other because he/she likes it more than the other. In instances where the user explicitly gives his/her preferences, such preferences can be directly modelled by a qualitative modelling framework such as the Strict Partial order framework. This cannot be directly done when preferences are

being implicitly learnt from user session data without a quantitative measure of how much a user prefers an item. So the challenge here is to define a measure of the degree to which a user likes an item. As discussed in Section 2.2.1, association rules and vector similarities cannot be directly used to define a preference measure. The most popularly used preference measures: *scores* and *frequencies* (or *probability*) of selection do not give an intuitive measure of a preference. Against this background, the discussion that follows illustrates how frequency (or probability) of selection fails to give an intuitive preference measure.

Consider a universal set,  $\xi$  of all item occurrences for a given user in a given domain; the set,  $U$  of items selections (viewings) by the same user in the same domain; and occurrence sets  $X_1, X_2, X_3$  and  $X_4$ , of items  $x_1, x_2, x_3$  and  $x_4$  in the universal set respectively. Refer to Figure 3.1.

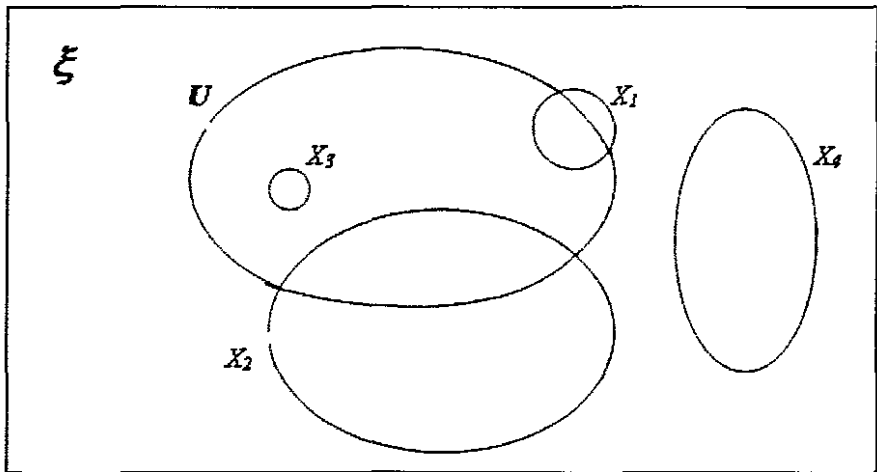


Figure 3. 1: Item accessibility, probability of selection, and preferences

Figure 3.1 depicts that the viewings (selections) of  $x_1$  in the user session,  $(X_1 \cap U)$  is less than that of  $x_2$  in the user session,  $(X_2 \cap U)$ . Thus, if frequency (or probability) of selection is used as a preference measure on items  $x_1$  and  $x_2$ , item  $x_2$  will be taken as more preferred to item  $x_1$ . However, relative to the occurrence of the items  $x_1$  and  $x_2$  in the universal set, it can be deduced that  $x_1$  was selected by the user almost half of the times it occurred and  $x_2$  was selected far less than a third of the time it occurred. This implies that item  $x_1$  is preferred to item  $x_2$ . It should be noted that  $x_1$  was selected less often than  $x_2$  because it is rarely distributed (not easily accessible) in the universal set. It is obvious that  $x_3$  is the most preferred item since it was selected each time there was a hit on it and  $x_4$  is the least preferred since in spite of its high occurrence the user never selects it.

From the foregoing discussion and the illustration in Figure 3.1, it is clear that an intuitive measure of a preference on a given set of items, in a given domain should consider the probability of the item being selected relative to the accessibility of the items in question. i.e.

$$pref(x) = \left( \frac{P(x;U)}{P(x)} \right) \quad 3.1$$

where  $P(x;U)$  is the probability of item  $x$  being selected and  $P(x)$  is the accessibility probability of item  $x$ .

### 3.1.3 Our User Preference Model

As proposed in (Jung et al, 2005), the measure of a preference on an item is calculated from the preferences of the item's features. The challenge that now arises is how the accessibility probability of a feature can be measured. To address this challenge we adapt the Closed World Assumption (CWA) introduced in (Kiessling, 2002). The Closed World Assumption (CWA) states that *the user knows of all possible values of all the choices that he/she can make*. Unlike the way the CWA was used in (Kiessling, 2002) to detect negative preferences, in this work it is used to define the accessibility probability of a feature.

In domains where the CWA holds, the accessibility probability of a feature,  $w$ , to a given user is given by the following equation:

$$P_A(x(w)) = \frac{n(A(w))}{n(A)} \quad 3.2$$

where  $n(A(w))$  is the total number of items with feature  $w$  in domain  $A$ , and  $n(A)$  is the total number of items in domain  $A$ . Equation 3.2 says the accessibility of a feature in the product catalogue or service registry can be generalised to all the users since they are all aware of its existence.

In domains where the CWA can not be assumed, the accessibility of an item to a given user is now dependent on whether the user is aware of its existence or not. In this case the accessibility of an item, and hence its features, differ for different

users. Equation 3.3 estimates the accessibility probability of a feature,  $w$ , to a given user in domains where the Closed World Assumption does not hold.

$$P_A(x(w)) = \frac{freq_{A,R}(x(w);U)}{freq_{A,R}(U)} \quad 3.3$$

where  $freq_{A,R}(x(w);U)$  is the frequency of hits/logs/recommendations of items with feature  $w$  in domain  $A$  for a given user,  $U$  and  $freq_{A,R}(U)$  is the total number of item hits for a given user,  $U$ , in domain  $A$ .

Another important concept this work defines is the concept of a Strong Negative Preference. An item is said to be a Strong Negative preference if the user is aware of its existence but he/she never selects it. In Table 3.1 we give definitions and notation of the parameters to be used in subsequent sections to get the preference measures to be used in this study.

## 3.2 User Preference Mining

The actual user preferences will be predicted from implicit preferences hidden in the user session data. Data mining techniques will be used to extract the preferences from the user session data. The mining of the user preferences will involve three major stages. These are:

1. *Mining* of attribute value preferences from the user session data;
2. *Matching* items' features and the user's feature preferences to compute the user's preferences on an item , and

### 3. Representation of preferences as Strict Partial Order relations.

Table 3. 1: Notation and Definitions

| Notation             | Definition   |
|----------------------|--|
| $n(A(w))$            | Number of items with feature $w$ in the in domain $A$ .  |
| $n(A)$               | Number of all items in domain $A$ .  |
| $freq_{A,R}(x,U)$    | The frequency item $x$ was recommended to a given user for selection in domain $A$ including cases were the user did not select it (i.e. number of hits on item $x$ in domain $A$ ).   |
| $freq_{A,S}(x,U)$    | The frequency item $x$ was selected by the user in domain $A$ .  |
| $freq_{A,R}(U)$      | Total number of item recommendations to a given user in domain $A$ .   |
| $freq_{A,R}(x(w);U)$ | Frequency of recommendation of items with feature $w$ to a given user in domain $A$ .  |
| $freq_{A,S}(x(w);U)$ | Frequency of selection of items with feature $w$ by a given user in domain $A$ .   |
| $freq_{A,S}(U)$      | Total number of item selections by a given user in domain $A$ .  |
| $P_A(x(w))$          | $n(A(w))/n(A)$ , measures accessibility probability of feature $w$ in domain $A$ , where the CWA can be assumed<br>OR<br>$freq_{A,R}(x(w);U)/freq_{A,R}(U)$ , measures the accessibility probability of a feature $w$ in domain $A$ , where the CWA can not be assumed |
| $P_A(x(w),U)$        | $freq_{A,S}(x(w);U)/freq_{A,S}(U)$ , measures the probability that a given user selects an item with feature $w$ in domain $A$ .   |
| $pref_A(w)$          | $\ln \left( \frac{P_A(x(w);U)}{P_A(x(w))} \right)$ , measures the preference of feature $w$ .  |
| $pref_A(x)$          | $\frac{1}{M(x)} \sum_{w_i \in x} pref_A(w_i)$ , measures the user's preference on item $x$ .   |

#### 3.2.1 Mining Attribute Value Preferences

Mining of attribute value preferences is divided in to two components, which are mining of attribute value preferences in continuous domains and mining of attribute value preferences in discrete domains. Each of these components is briefly discussed below.

### A. Mining Attribute Value Preferences in Categorical Domains

The parameters defined in Table 3.1 are used to mine for preferences on categorical domains. The preference on a given feature is given by the mutual information

$$pref_A(w) = \ln \left( \frac{P_A(x(w); U)}{P_A(x(w))} \right) \quad (3.3)$$

where the  $P_A(x(w), U)$  and  $P_A(x(w))$  defines the selection probability of an item with feature  $w$  and the accessibility probability of an item with feature  $w$  in domain  $A$  respectively.

### B. Mining Attribute Value Preferences in Continuous Domains

Continuous attributes need a different approach to data driven preferences. This is because the domain for continuous domains contains an infinite number of values. In this case one value occurs once or just a few times. In such situations the approach for mining preferences in discrete domains does not work, since it is based on preferences on values. (Holland, 2003) highlighted two issues about numeric data that can be exploited in order to mine preferences in continuous domains:

- Continuous data is ordered. This additional knowledge should be used for preference mining, and

- Statistics has lots of tools for analysing continuous data (density estimation, histograms, etc). Possibly some are appropriate for preference mining task.

The distribution of the continuous numeric values defines a probability density function  $f(x)$ . The major challenge here is that the density function is not known a priori and therefore it has to be estimated. This work adopts the use of histograms to estimate the underlying density function. The distribution of the values in the continuous domain is used to estimate the accessibility of the values in a given range of values. The selection probability of the values in a given range will be estimated using the values that are in the log relation data.

### 3.2.2 Computing Item Preference Values

Preferences on the items are obtained by matching the items' features and the user's attribute value preferences (both for attributes in discrete and continuous domains) using the equation:

$$pref_A(x) = \frac{1}{M(x)} \sum_{w_i \in x} pref_A(w_i) \quad (3.4)$$

where  $M(x)$  is the number of common attributes for items in domain  $A$ . This approach was chosen over the approach of aggregating preferences using prioritised and Pareto preference constructors as proposed in the preference modelling framework in (Kiessling, 2002) after the following consideration:

*The strict partial order preference framework as presented in (Kiessling, 2002) was designed for situations where the user explicitly specifies his*

*preferences and the order in which they are to be considered and hence the complex preference constructors were designed accordingly. In this case it will not be appropriate to apply such constructors for automatically analysing preference models without human intervention.*

However, apart from the gaps left out by the strict partial order preferences framework (Kießling, 2002) and the subsequent preference mining framework in (Holland et al, 2003), we adopt the concept of Data Driven Preferences, denoted by  $P_D = (A, \prec_{PD})$ , from (Holland et al, 2003). This concept forms the basis for extraction of user preferences from user session data in our work. Holland et al (2003) defines the following data driven preferences:

- For preference in a categorical(discrete) domain ,  $dom(A)$ , a data-driven preference  $P_D = (A, \prec_{PD})$  is defined as:  $x \prec_{PD} y$  iff  $freq_A(x) < freq_A(y)$
- For preferences in a continuous domain,  $dom(A)$ , data driven preference  $P_D = (A, \prec_{PD})$  is defined as  $x \prec_{PD} y$  iff  $\exists \varepsilon > 0: freq_A([x - \varepsilon, x + \varepsilon]) < freq_A([y - \varepsilon, y + \varepsilon])$

Unlike what was proposed in (Kießling, 2002), our research work's approach does not need representation of preferences at feature level as Strict Partial Orders. Only preferences on items are represented as strict partial orders and hence no need to consider representation of strict partial order preferences in continuous domains. Replacing the frequency of selection with the value of  $pref_A(x)$  as a measure of a preference, we define a data driven preference on an item as follows:

- For  $\forall x, y \in \text{dom}(A)$ , a data-driven preference  $P_D = (A, \prec_{PD})$  is defined as:

$$x \prec_{PD} y \text{ iff } \text{pref}_A(x) < \text{pref}_A(y)$$

Using the above mentioned definition of a data driven preference on an item, the following Classes of preferences are defined:

- An item  $x$  is a Strong Negative (STRONG\_NEG) preference in domain  $A$

$$\text{iff } \text{freq}_{A,R}(x, U) > \text{freq}_{A,R}(\text{threshold}) \text{ and } \text{freq}_{A,S}(x; U) = 0$$

- An item  $x$  is a Soft-Negative (SOFT\_NEG) preference in a given domain if

$$\text{pref}_A(x) < 0 .$$

- Item  $x$  is a Positive (POS) preference in domain  $A$  only if  $\text{pref}_A(x) \geq 0$

where  $\text{freq}_A(\text{threshold})$  is threshold frequency item  $x$  must have been made available to the user for selection in  $\text{dom}(A)$  for it to be considered to have been recommended sufficiently enough for the user to be aware of it.

The positive and soft-negative preferences classes defined above form a bipolar preference scale with a preference value of zero being the pivot point. Items with preference values less than zero represents the disliked items and those greater or equal to zero representing the liked items. This allows for representation of negatively preferred items as strict partial order relations, which the Strict Partial order framework presented in (Kießling, 2002) in cannot do. The work done in (Sicilia and Garcia, 2004) found that the consideration of negative preference measures (i.e. bipolar preference modelling approaches) as inhibitors of the recommendation process may be more appropriate to reduce false positives.

### 3.2.3 Strict Partial Order Representation of Preferences

Using the definition of preference classes given above the following strict partial order data driven preferences are defined:

- There is a data-driven POS preference, iff  $\forall x \in POS\text{-}set, \forall y \notin POS\text{-}set$   
 $: y \prec_{PD} x$ .
- There is a data-driven STRONG\_NEG preference, iff  
 $\forall w \in STRONG\_NEG\text{-}set, \forall y \notin STRONG\_NEG\text{-}set : w \prec_{PD} y$ .
- There is a data-driven SOFT\_NEG preference, iff  
 $\forall x \in SOFT\_NEG\text{-}set, \forall w \in STRONG\_NEG\text{-}set, \forall z \notin (SOFT\_NEG\text{-}set \cup STRONG\_NEG\text{-}set):$   
 $w \prec_{PD} x, x \prec_{PD} z$ .
- There is a data-driven POS/POS preference, iff  $\forall x \in POS1\text{-}set,$   
 $\forall y \in POS2\text{-}set, \forall z \notin (POS1\text{-}set \cup POS2\text{-}set): z \prec_{PD} y, y \prec_{PD} x$ .
- There is a data-driven SOFT\_NEG/SOFT\_NEG preference, iff  $\forall x \in$   
 $SOFT\_NEG1\text{-}set, \forall y \in SOFT\_NEG2\text{-}set, \forall w \in STRONG\_NEG\text{-}set, \forall z \notin (SOFT\_NEG1 \cup$   
 $SOFT\_NEG2\text{-}set \cup STRONG\_NEG\text{-}set): w \prec_{PD} y, y \prec_{PD} x, x \prec_{PD} z$ .
- There is a data-driven POS/SOFT\_NEG preference, iff  
 $\forall x \in POS\text{-}set, \forall y \in SOFT\_NEG\text{-}set, \forall w \in STRONG\_NEG\text{-}set, \forall z \notin (POS\text{-}set \cup SOFT\_NEG\text{-}set$   
 $\cup STRONG\_NEG\text{-}set : w \prec_{PD} y, y \prec_{PD} z, z \prec_{PD} x$ .
- There is a data-driven EXPLICIT\_SOFT\_NEG preference, iff  
 $\forall y_1 \in CATEGORY_{SN}1\text{-}set, \forall y_2 \in CATEGORY_{SN}2\text{-}set \dots \forall y_m \in CATEGORY_{SN}m\text{-}set,$   
 $\forall w \in STRONG\_NEG\text{-}set : y_2 \prec_{PD} y_1, y_3 \prec_{PD} y_2 \dots y_m \prec_{PD} y_{m-1} \text{ and } w \prec_{PD} y_m$ .

- There is a data-driven EXPLICIT\_POS preference, iff  
 $\forall x_1 \in \text{CATEGORY}_{p1\text{-set}}, \forall x_2 \in \text{CATEGORY}_{p2\text{-set}} \dots \forall x_n \in \text{CATEGORY}_{pn\text{-set}},$   
 $\forall w \in \text{STRONG\_NEG-set} : y_2 \prec_{PD} y_1, y_3 \prec_{PD} y_2 \dots y_m \prec_{PD} y_{m-1} \text{ and } w \prec_{PD} y_m.$
- There is a data-driven POS/EXPLICIT\_SOFT\_NEG preference, iff  $\forall x \in \text{POS-set},$   
 $\forall y_1 \in \text{CATEGORY1-set}, \forall y_2 \in \text{CATEGORY2-set} \dots \forall y_n \in \text{CATEGORYN-set},$   
 $\forall w \in \text{STRONG\_NEG-set} : y_1 \prec_{PD} x, y_2 \prec_{PD} y_1, y_3 \prec_{PD} y_2 \dots y_n \prec_{PD} y_{n-1}, \text{ and}$   
 $w \prec_{PD} y_n.$
- There is a data-driven EXPLICITPOS/SOFT\_NEG preference, iff  
 $\forall y_1 \in \text{CATEGORY 1-set}, \forall y_2 \in \text{CATEGORY 2-set} \dots \forall y_n \in \text{CATEGORYN-set},$   
 $\forall x \in \text{POS-set}, \forall w \in \text{STRONG\_NEG-set} : y_2 \prec_{PD} y_1, y_3 \prec_{PD} y_2 \dots y_n \prec_{PD} y_{n-1},$   
 $x \prec_{PD} y_n \text{ and } w \prec_{PD} x.$
- There is a data-driven EXPLICIT\_POS/EXPLICIT\_SOFT\_NEG preference, iff  
 $\forall x_1 \in \text{CATEGORY}_{p1\text{-set}}, \forall x_2 \in \text{CATEGORY}_{p2\text{-set}} \dots \forall x_n \in \text{CATEGORY}_{pn\text{-set}},$   
 $\forall y_1 \in \text{CATEGORY}_{SN1\text{-set}}, \forall y_2 \in \text{CATEGORY}_{SN2\text{-set}} \dots \forall y_m \in \text{CATEGORY}_{SNm\text{-set}},$   
 $\forall w \in \text{STRONG\_NEG-set} : x_2 \prec_{PD} x_1, x_3 \prec_{PD} x_2 \dots x_n \prec_{PD} x_{n-1}, y_1 \prec_{PD} x_n$   
 $y_2 \prec_{PD} y_1, y_3 \prec_{PD} y_2 \dots y_m \prec_{PD} y_{m-1} \text{ and } w \prec_{PD} y_m.$
- Let  $\prec_E$  a strict partial order on E. A data-driven EXPLICIT preference holds, iff
  - $\forall x, y \in E$  with  $x \prec_E y, x \prec_{PD} y.$
  - $\forall u \in E, \forall v \notin E : v \prec_{PD} u.$

The formalisation of the above defined preferences based on the Preference Algebra defined in (Kiessling, 2002) is shown in Appendix D.

### 3.3 Mining Context-based User preferences

The foregoing discussion of the user preference mining framework was an attempt to solve the first research challenge raised in this study which is: Finding ways of representing/modelling user preferences that gives *an intuitive interpretation of a preference* and *user expressiveness* to enhance user models for advanced personalisation in the provision of m-Services. This section discusses how our proposed user preference mining framework can extract context-based user preferences from user session data. There are quite a number of challenges that have to be addressed for this goal to be a reality. The key challenge is the need for a sound context model for an m-services environment. In this work we defined our envisaged context model. The evaluation of this context model is beyond the scope of this work and will be left as part of our future work. The model will only be justified on the basis of its complexity analysis. The following section discusses our proposed context model for a mobile computing environment.

3.3.1 Context Model for Mobile Services

Holland and Kiessling (2004) proposed an e-commerce context meta-model with three high level context components: location, time and influences. Due to the fact that in a mobile computing scenario the user might be involved in some other activities other than computing (e.g. driving, running, etc), the user's activity has to be considered to avoid recommendations that are irrelevant to the user's activity. Based on this consideration we came up with the context meta-model shown in Figure 3.2, in which a forth component, *activity* was added to *location*, *influences* and *time*.

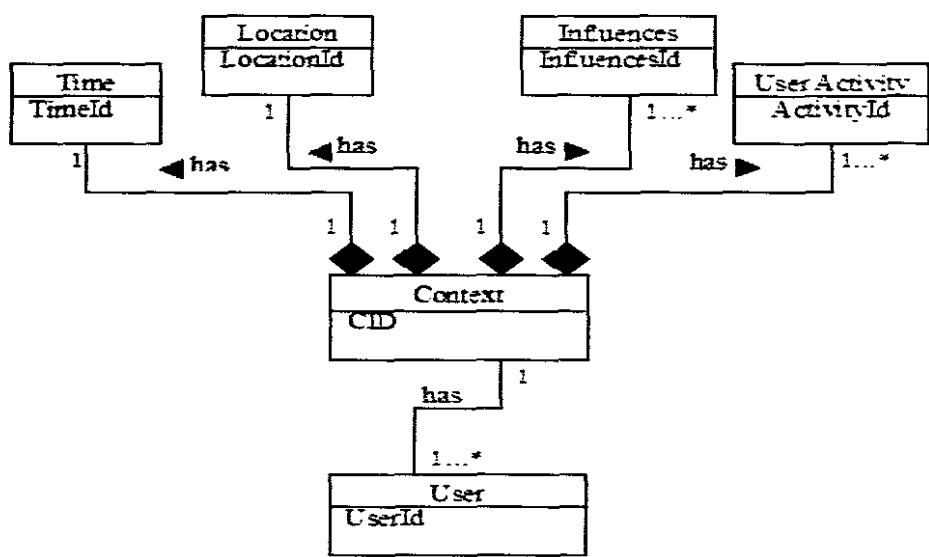


Figure 3. 2: Context Meta-Model for m-Services, an Adaptation of the Context Meta-Model from (Holland et al, 2004)

Considering the nitty-gritties of context each context occurs once or just a few times, which tends to discourage the whole idea of detecting context-based user preferences. Based on the context model for a context-based service registry in (Doukeridis et al, 2006), we develop a context model to support the mining and usage of context-based user preferences. Since we do not want each context description to be unique, context descriptions have to be clustered based on their similarities. Some incremental clustering algorithm will be defined to cluster context first relative to time, then location, activity and lastly influences. A Context Identity (CID) is assigned to every leaf of the tree structure shown in Figure 3.3. The tree structure applies a breath first search for the CID, which any given context description is most similar to.

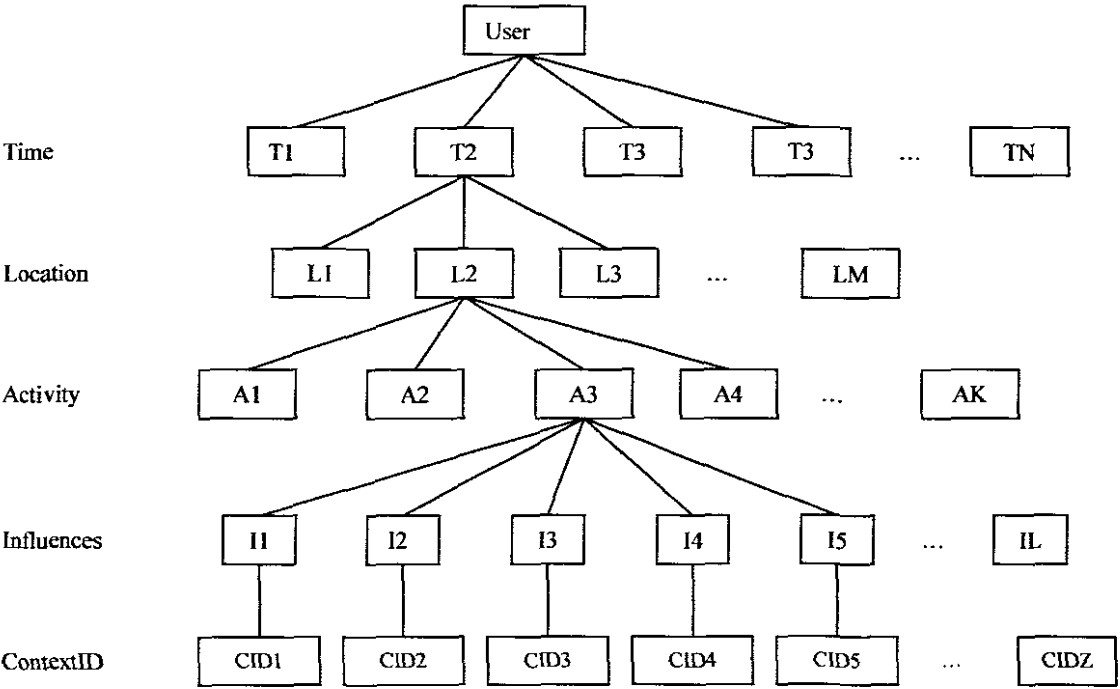


Figure 3. 3: Context modelling in an m-Services environment.

To get the CID most similar to the current context, the algorithm has to span at most  $N$  nodes for time,  $M$  nodes for location,  $K$  nodes for activity and  $L$  nodes for influences. This gives a linear  $(N+M+K+L)$  time complexity. The space complexity is also minimal since the algorithm only has to store only the node with the minimum distance from the current context description at each level.

### 3.3.2 Our Context Model and User Preference Mining

The personalised CID generated from the context modelling framework discussed in Section 3.3.1 will be used as an index for storing the user session data (from which the context-based user preferences are to be mined). Each time the user preference miner is invoked three parameters are passed to it; the *current context (CID)* generated from the framework discussed in Section 3.3.1, *userID* (unique user identifier) and the *domain* the user is interested in. Thus each time the preference miner is invoked, it returns preferences for items in a given domain relevant to the current context.

## 3.4 User Preference Mining Algorithms

This section presents algorithms that are to be used to extract context-based user preferences based on the preference model and the preferences mining framework presented in Section 3.1 and Section 3.2. The proposed user preference mining algorithm consists of four parts namely:

1. Mining of item attribute value preferences;
2. Matching items' features and the user's feature preferences to compute the user's preferences on an item;
3. Mining for SPO categorical preferences, and
4. Mining for explicit preferences within preference categories.

#### **A. Algorithm for Mining Attribute Value Preferences**

This algorithm takes the userID, CID and the Domain of the items whose preference are to be mined as input. The user input is first used to compute the selection and accessibility probabilities for all item features in that Domain, both in discrete and continuous domains. These probabilities are then used to compute the user's preferences on each of the attribute values (features). Equation 3.3 is used to compute the attribute value preferences from the accessibility and selection probabilities. This algorithm returns a vector of the user's feature preferences in a given domain (Domain) under a given context (CID). The algorithm is shown in Figure 3.4. This algorithm runs at the database layer.

#### **B. Algorithm for Matching items' features and the user's feature preferences**

The algorithm for matching the user's feature (attribute value) preferences and the items' features take the vector of feature preferences from the attribute value preference mining algorithm and the item profiles as input (see Figure 3.5).

**Input:** UserID, CID, Domain.

1. get all the attributes in discrete domains under the current Domain
2. for a given UserID:
  - a. compute the accessibility probability of each feature value,  $w_i$ , under the current context (CID).
  - b. compute the selection probability of each feature value,  $w_i$ , using the user history under the current context (CID).
3. get all the attributes in continuous domains under the current Domain
4. for each attribute estimate the density function using its attribute values in the current Domain.
5. For a given UserID:
  - a. estimate the accessibility probability of each range of values,  $w_i$ , under the current context (CID) using the density function estimated by (4)
  - b. estimate the selection probability of each range of values,  $w_i$ , under the current context (CID) based on the density function estimated by (4)
6. For a given UserID, and Domain under the current context (CID) compute the attribute value preferences,  $pref_A(w_i)$ , for attributes both in discrete and continuous domains

**Output:** User Preference Profile =  $\{ pref_A(w_1), pref_A(w_2), pref_A(w_3), \dots, pref_A(w_m) \}$

Figure 3. 4: Algorithm for mining preferences on item features/attribute values

**Input:** User Preference Profile (Vector), item profiles (Vector)

for each item ( $x_i$ ) in the item profiles (Vector) get its attributes and attribute values

- ```

{
  1. for each entry in the User Preference Profile
    {
      if the attribute value in the product profile is equal to attribute value in the User Profile
        {
           $pref(x_i) = pref(x_i) + \text{attribute value preference}(pref_A(w_i))$ 
        }
    }
  2. Normalise  $pref(x_i)$ 
}

```

**Output:** ItemPrefValues =  $\{ pref(x_1), pref(x_2), pref(x_3), \dots, pref(x_i) \}$

Figure 3. 5: Algorithm matching Items' features to the users attribute value preferences

The algorithm matches product's features to the attribute value preferences and compute the preferences on an item by summing the attribute value preferences of the features of an item and normalise them by dividing the 'sums' by the number of features the item has. This then gives the user's preference value on a given item. Equation 3.4 is used to compute the user's preference values for each and every item in a given domain. The algorithm outputs a vector of item preferences under the current context in a given domain.

### **C. Algorithm for mining SPO Categorical Preferences**

The algorithm for mining Strict Partial Order categorical preferences on items (shown Figure 3.6) takes the output from the matching algorithm, plus the UserId, the context (CID) and the domain. The algorithm first identifies the STRONG\_NEGset and removes it from the set of items to be clustered. After removing the STRNG\_NEGset the algorithm then invokes the clustering algorithm, first, to cluster the POS preferences and then second, to cluster the SOFT\_NEG preferences. The POS and SOFT\_NEG preferences are clustered separately to avoid cases where the clustering algorithm puts SOFT\_NEGs and POSs in the same cluster. The algorithm outputs strict partial order data driven categorical preferences (Note that categorical preferences were defined in Section 3.2.3). Since our preference mining algorithms will be running online, hierarchical clustering algorithms will tend to take more time in the process of synthesising SPO categorical preferences due to their high time and space complexities (see Table 2.2). Incremental clustering algorithms return

inconsistent clusters for each run depending on the order in which the processing of samples was done. Partitional clustering algorithms work best for our purposes. This is because of their relatively low time and space complexities and relatively high consistency in the returned clusters. Against this background the k-means partitional clustering algorithms is used in this study. The silhouette technique (Holland, 2003) is adapted to determine the best possible clustering.

#### **D. Algorithm for Mining Explicit Preferences within Preference Categories**

Within preference categories the user preferences will be represented as EXPLICIT preference (See definition in Section 3.2.3) where applicable. The algorithm for mining explicit preferences adapted from (Holland, 2003) is shown in Figure 3.7. This algorithm takes as input the highest preference category from the SPO Categorical preference mining algorithm and some log relations  $LR(LogID, UserID, ContextID, SessionID, ItemID, Attribute\ value, selected)$  for the items in the same preference category corresponding to a given user. For each instance the algorithm is invoked, it initialises an empty Directed Acyclic Graph (DAG). Then it does preference comparisons of each item in that category to each of the items in the same category with it. The comparison is done through checking that for all the sessions two items appeared together, which one was selected more than the other. If an item,  $x$ , is found to have being selected more times than item,  $y$ , then a relation  $x$  is better than  $y$  (i.e.  $x$  is preferred to  $y$ ) is added to the DAG.

**Input:** UserID, CID, domain, ItemPrefValues = {pref(x<sub>1</sub>), pref(x<sub>2</sub>), pref(x<sub>3</sub>), ... , pref(x<sub>n</sub>)}.

1. For each  $x_i$  in a given domain,  $A$ , under the current context compute  $freq_A(x_i; U)$  and  $freq_A(x)$
2. Remove all  $x_i$ :  $freq_A(x_i; U) = 0$  and  $freq_A(x_i) \geq freq_A(threshold)$  from the item set. This is a set of Strong Negative (STRONG\_NEG) preferences.
3. Extract the  $pref_A(x)$  values for the remaining items
4. Compute a clustering of the  $x_i$ 's with  $Pref_A(x) \geq 0$  and that of the  $x_i$ 's with  $Pref_A(x) < 0$  separately, using a clustering technique

Depending on the clustering results we have the following possibilities:

a. One cluster,  $C_1$ , we have a:

- i. SOFT\_NEG( $A, C_1; \{x \in dom(A) | Pref_A(x) < 0\}$ ),
- ii. Pos( $A, C_1; \{x \in dom(A) | Pref_A(x) \geq 0\}$ )

b. Two clusters,  $C_1$  and  $C_2$ , where  $\forall c_1 \in C_1, \forall c_2 \in C_2, Pref_A(c_2) < Pref_A(c_1)$ . We have a:

- i. POS/SOFT\_NEG( $A, C_1; \{x \in dom(A) | Pref_A(x) \geq 0\}; C_2; \{x \in dom(A) | Pref_A(x) < 0\}$ )
- ii. POS/POS ( $A, C_1; \{x \in dom(A) | Pref_A(x) \geq 0\}; C_2; \{x \in dom(A) | Pref_A(x) \geq 0\}$ )
- iii. SOFT\_NEG/SOFT\_NEG( $A, C_1; \{x \in dom(A) | Pref_A(x) < 0\}; C_2; \{x \in dom(A) | Pref_A(x) < 0\}$ )

c. More than two clusters  $C_1, C_2, \dots, C_n$ , where  $\forall c_1 \in C_1, \forall c_2 \in C_2, \dots, \forall c_n \in C_n$ :

$Pref_A(c_n) < \dots < Pref_A(c_2) < Pref_A(c_1)$ . We have :

- i. an EXPLICIT\_POS preference  $E(A, \prec_{EP})$  with  $c_n \prec_E c_{n-1} \prec_E \dots \prec_E c_1, Pref_A(x) \geq 0, \forall c_i, i = 1 \dots n$ .
- ii. an EXPLICIT\_SOFT\_NEG preference  $E(A, \prec_{EP})$  with  $c_n \prec_E c_{n-1} \prec_E \dots \prec_E c_1, Pref_A(x) < 0, \forall c_i, i = 1 \dots n$
- iii. an EXPLICIT\_POS/SOFT\_NEG preference  $E(A, \prec_{EP})$  with  $c_n \prec_E c_{n-1} \prec_E \dots \prec_E c_1$ :  $c_n \in dom(A) | pref_A(c_n) < 0, c_i \in dom(A) | pref_A(c_i) \geq 0, \forall i = 1, 2, 3 \dots n-1$ .
- iv. a POS/EXPLICIT\_SOFT\_NEG preference  $E(A, \prec_{EP})$  with  $c_n \prec_E c_{n-1} \prec_E \dots \prec_E c_1$ :  $c_1 \in dom(A) | pref_A(c_1) \geq 0, c_i \in dom(A) | pref_A(c_i) < 0, \forall i = 1, 2, 3 \dots n-1$ .
- v. an EXPLICIT\_POS/EXPLICIT\_SOFT\_NEG preference  $E(A, \prec_{EP})$  with  $c_n \prec_E c_{n-1} \prec_E \dots \prec_E c_1$ :  $c_i \in dom(A) | pref_A(c_i) \geq 0, \forall i = 1, 2 \dots m, c_i \in dom(A) | pref_A(c_i) < 0, \forall i = m+1, m+2 \dots n$

5. In all the other cases there are no data-driven preferences

**Output:** The detected categorical preferences or no preference was found

Figure 3. 6: Algorithm for mining categorical preferences

**Input:** log data: LR(LogID, UserID, ContextID, SessionID, ItemId, Attribute value, selected) for a given preference category; highest preference category from the SPO preference mining algorithm.

1. Compute the  $k$ -occurring values  $(x_1, x_2, \dots, x_k)$  in the log relation. Initialise the better than graph with  $E - graph = \emptyset$
2. *FOR* ( $i=1, \dots, k$ ) and *FOR* ( $j=i+1, \dots, k$ ) *DO*:
  - a. Consider the sessionIDs, whose according values contain  $x_i$  and  $x_j$ .
  - b. Compute,  $s$ , the number of SessionIDs, where  $x_i$  was selected and  $x_j$  wasn't
  - c. Compute,  $t$ , the number of SessionIDs, where  $x_j$  was selected and  $x_i$  wasn't
  - d1. If  $s > t$ , set  $E-graph = E - graph \cup (x_i, x_j)$ .
  - d2. If  $s < t$ , set  $E-graph = E - graph \cup (x_j, x_i)$ .
  - d3. If  $s=t$  do nothing.
3. Remove all circular paths by removing edges to the item with the highest out degree in the circular path.

**Output:** the detected EXPLICIT preferences represented in a DAG

Figure 3. 7: Algorithm for mining explicit preferences

Circular path are removed by removing all the edges to the item with the highest out degree in the cycle. The algorithm outputs EXPLICIT preferences represented in a DAG. In this case all the items in the highest preference category will be in the DAG and are thus detected to be more preferred to all the items in the lower categories, which fulfils the fact that all the items in the DAG are more preferred to all the items not in the DAG in the definition of EXPLICIT preference presented in Section 3.2.3.

The logic behind the explicit preferences stems from the fact that since preferences are deduced from the features which are common to items in a given

domain, items with almost similar features will have preference values close to each other and hence are most likely to be in the same preference category. To get which one is more preferred to the others, we need to compute relative preferences of each item to all the other items in the same preference category with it. The irreflexive, asymmetric and the transitive properties of strict partial order relations are used to get an explicit representation of the preferences of all the items in that category.

### 3.5 User Modelling for m-Services

User preferences mining framework in the foregoing discussion is meant for mining user preferences for supporting advanced personalisation in Mobile Services. A typical way of achieving this is by incorporating user preferences into a user model, which in turn is used to personalise mobile services

We take inspiration for user modelling from the work done by (Kay et al, 2001).

The architecture was selected because of the following reasons:

- It supports distributed context based personalisation, and
- Allows different applications to share their user models (Cross Personalisation).

This work proposes the user modelling architecture for an m-services environment shown in Figure 3.8. The Capital letter U represents the Definitive User Model. Each application, A, has its own Partial User Model, u. The ovals

marked I represent inference engines, the internal reasoning mechanism for the user models. The applications can share user models through the Definitive User Model which will be held in the middleware as shown both in Figure 3.8 and Figure 3.9. The Definitive User Model will also be responsible for handling user preferences for automatic service discovery, selection and execution.

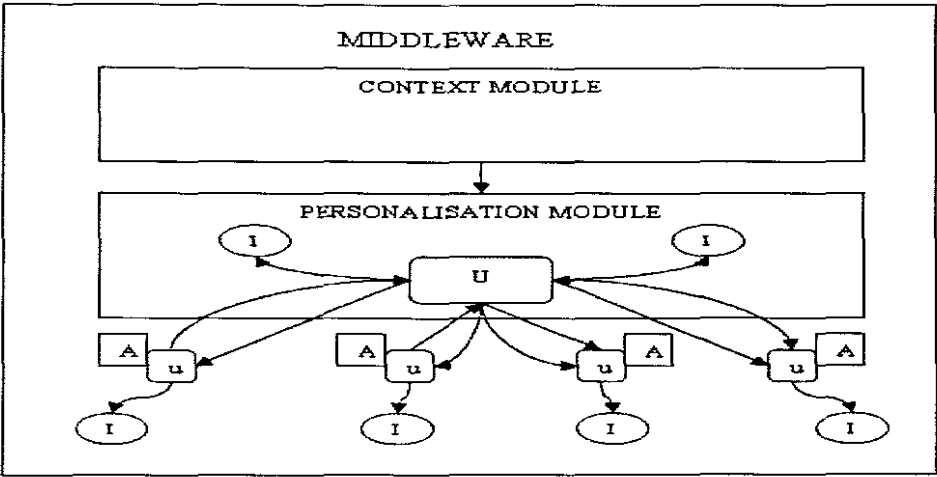


Figure 3. 8: User Model Architecture for Personalisation in m-Services

### 3.6 Infrastructure for User Preference Mining and Usage in m-Services

It follows from the foregoing discussion that a mechanism is required to put the various components developed in this research together. To this effect, this Section presents an infrastructure for user preference mining and usage in an m-

services environment we named User Preference Centred Architecture for Mobile Services (UPCAMS).

Following the discussion presented in Section 2.3.2, user preferences can be mined and used at two levels in an m-services environment. The first one, which is at middleware level, involves mining of user preferences from sequential service access data in the form of web logs. The preferences mined at this level are on the m-services and will be used for automatic service discovery, selection and composition. Based on the preference model defined above there is a need for an enriched service description, which clearly specifies the attributes/features for each and every m-service available. The second one involves user preference mining from web logs from the application servers within applications such as e-Commerce applications, map services, restaurant finders, etc. In this section we will start by presenting the design principles of UPCAMS in Section 3.6.1. In Section 3.6.2 we present UPCAMS and its components

### **3.6.1 UPCAMS Design Considerations**

The key design considerations which influenced the designing of our framework for user preference mining are: separation of context and personalisation in mobile computing, capacity scalability of the user preference mining architecture, and reusability of software and hardware components.

### 3.6.1.1 Separation of context and personalisation

We have demonstrated in Section 2.2, that context-aware computing takes personalisation information as context information. This results in personalisation information such as user preferences being taken as part of context data. Ideally in context-aware computing, all context abstraction processes are handled in the middleware. Applications subscribe, poll or query for relevant context from the middleware. In a bid to effectively use the techniques that have emanated from both the fields of context-aware computing and personalisation, we have in this research separated personalisation from context based information as discussed in Section 2.1 and later reflected in this Sub-section 3.6.2 in the architecture developed in this work.

### 3.6.1.2 Capacity Scalability

Scalability of a system indicates the ability of a system to scale up with the growing amounts of work. Since the proposed data mining algorithms will be working on large amounts of data, it is important that they scale up with the large amounts of data. To this end, this research work is so particular on the choice of the data mining algorithms to be used. The user preference mining framework presented in Sections 3.2, 3.3 and 3.4 are designed in such a way that they will scale up with increase with in the number of tuples in the underlying data warehouse. Only the data access is affected by the increase in the number of tuples in the database. The DAG will only be expensive when the clustering

algorithm has failed to find a reasonable structure among the items to be clustered. The matching and the clustering algorithm do not depend on the number of tuples in the data warehouse. The major scalability drawback on the clustering algorithm is in the number of items in a given domain. This is taken care of by the fact that our clustering algorithm runs on one-dimensional data, which lessens the time it takes to cluster the items. Thus our preference miner will be much faster than vector similarity based preference models, the content based and collaborative filtering based models, since they deal with running through and clustering multi-dimensional data.

### 3.6.1.3 Reusability and Sharing of Software and Hardware

#### Components among Applications

This is attained through modular programming and through the emerging web services technology. The need for modular programming makes JAVA a natural solution for implementing a prototype of the proposed user preference mining framework. This is because of JAVA's ability to support reusability, rapid prototyping and easy integration of existing modules. Modular solutions bring advantages of reusability and extensibility among other advantages. To simplify the task of creating, maintaining, and extending context-aware m-services and to support collaboration among similar applications much of the weight of context-aware computing and personalisation must be shifted to the middleware. The implementation of the preference miner as a service also serves this purpose.

### 3.6.2 UPCAMS and its Components

This research work proposes architecture for mining and usage of user preferences in an m-Services environment shown in Figure 3.9. The architecture is divided into three layers, the User Interface, Middleware, and Application layers where user preferences can be used, but preference mining is only done in the Middleware and the Application layer. Each of these two layers has its own components to support the mining and usage of user preferences, except for the context module, which is only in the Middleware layer. Applications access this module through polling, querying or subscribing for the context relevant to them, based on the nature of the application and/or the user's preferences. For instance, some applications support context triggered actions and hence they need to keep checking (through polling or subscribing for the relevant context) for context changes for all users who need to be notified of any context changes.

#### 3.6.2.1 Middleware Components

As discussed in Section 3.6.1 another very important design consideration for the efficiency of the framework proposed in this work is the separation between context and personalisation in the middleware layer. The middleware is divided into two modules the context and the personalisation modules based on the separation of personalisation and context discussed in Section 2.2.

### 3.6.2.1.1 Context Module

The context module provides the basic functionalities of a context-aware middleware such as context extraction, abstraction and modelling, except for the handling of personalisation information such as user preferences, interest, etc, which will be handled in the personalisation module. Applications subscribe, poll or query for context information from this module and the module will in turn configure some context provider components according to the application's requirements.

### 3.6.2.1.2 The Personalisation Module

The personalisation module holds the functionalities for personalisation required at middleware level. The module consists of four main components which are User Session Manager, Context History, Middleware User Session Data Warehouse, and the definitive user model.

#### **A. User Session Manager (USM)**

The user session manager is responsible for handling all interaction sessions in the personalisation module. It gets the context data from the context module and initiates some context triggered actions at middleware level.

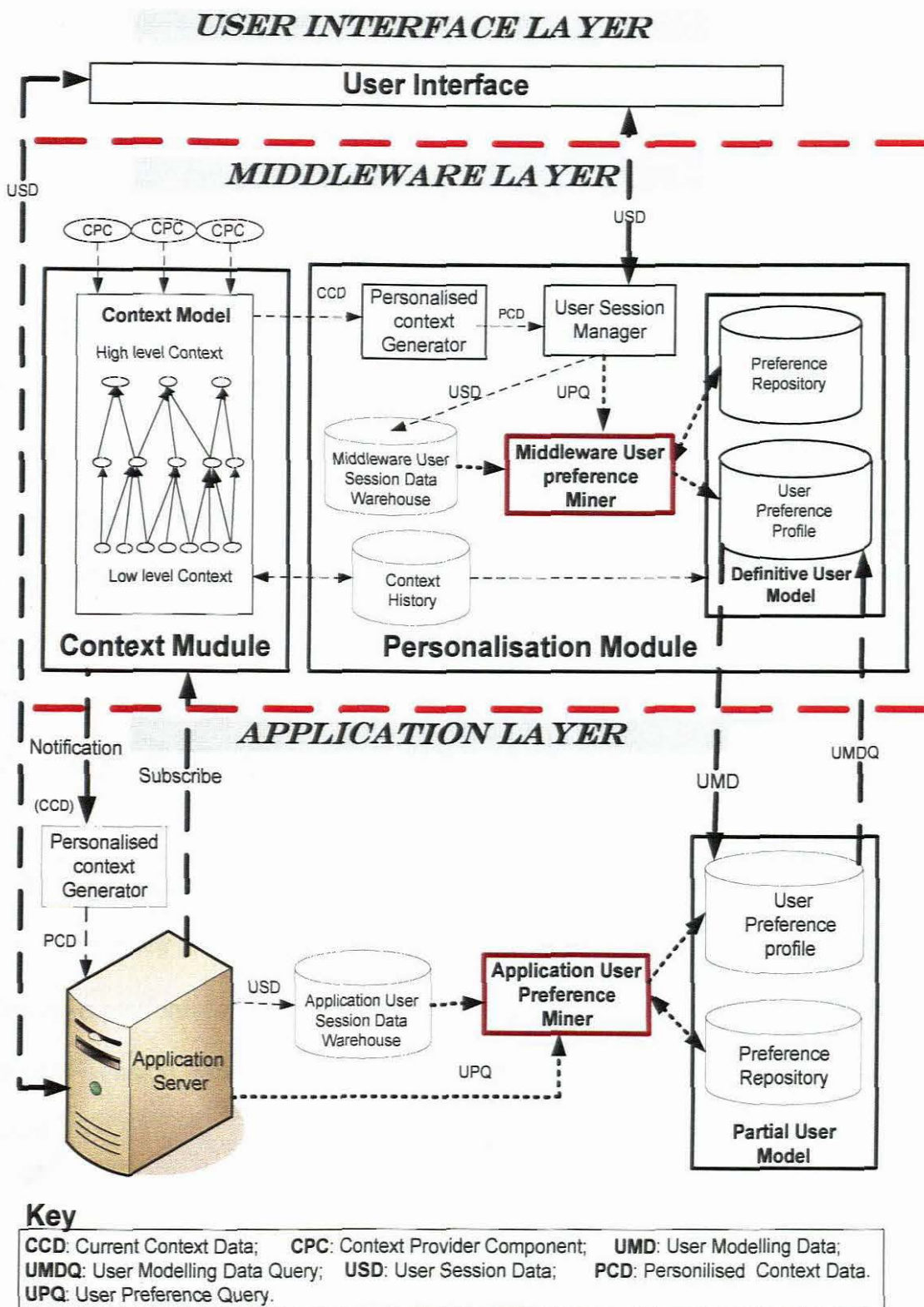


Figure 3. 9: User Preference Centred Architecture for Mobile Services (UPCAMS).

### **B. Personalised Context Generator**

This component is a data model responsible for personalising the context data from the context module and assigning it to a context cluster whose elements it is most similar to (see Section 3.3.1).

### **C. Context History**

This is a repository where context data is stored. The data from context histories can be used for user modelling and for higher level context abstraction. That is the context history can also act as a context provider component. The context history is placed in the personalisation module based on the definitions of context and personalisation given in this study. This is based on the fact that the context history is meaningless if not attached to a specific user. This decision also eases the management of context histories.

### **D. Middleware User Session Data Warehouse (MUSDW)**

This component is a persistent repository for user session data, responsible for creating a platform for user preference mining in the middleware layer. It holds all the user session data and the details of the items whose preferences are to be mined.

### **E. Definitive User Model (DUM)**

The Definitive User Model holds all the personalisation data needed at middleware level including the user preferences mined from the middleware user

session data warehouse and the rules on how the data is going to be used for personalisation. The user preferences are queried autonomously based on the current context and used to recommend services to the user. The definitive user model is also responsible for facilitating sharing of User Preference Profiles among similar applications.

### **3.6.2.2 Application Components**

The application Layer has 5 main components. These are the Application Server, Personalised Context Generator, Application User Session Data Warehouse, Application User Preference Miner and the Partial User Model.

#### **A. Application Server (AS)**

Wireless internet service applies application servers to execute the service business logic and hosts the database that stores web content (Pashtan, 2005). Typical services provided by the application server infrastructure include multiple processing to handle multiple client requests and simultaneous backend database queries, client session management, page caching and data streaming. Taking advantage of their user session management capabilities our user session data is going to be drawn from the application server. The application server will also be responsible for polling, querying, or subscribing for context data from the context module.

#### **B. Personalised Context Generator (PCG)**

This functions the same way as the PCG in the middleware layer. Before the context description can be used in the personalisation process it has to be personalised and assigned to a context cluster with a unique identifier (See Section 3.3.1) based on its similarity to the context in that cluster.

#### **C. Application User Session Data Warehouse (AUSDW)**

The Application User Session Data Warehouse is a persistent database that provides a platform for user preference mining in the application layer. It holds all the user session data at application level and the details of the items whose user preferences are to be mined.

#### **D. Partial User Model (PUM)**

This is a repository that holds all the personalisation information needed at application level and the rules on how it is going to be used in the personalisation process. Preferences mined from the Application User Session Data Warehouse are stored in this repository. The preferences are queried based on the current context and used to recommend items to the user. Partial User Models of similar applications can share User Preference Profiles and other user modelling data through the Definitive User Model.

### 3.6.3 The User Preference Mining Component

Ideally the data mining component should be implemented as a service that can be invoked, in the middleware or by different m-services applications to synthesise reasonable preferences and store them in the respective Preference Repositories. Further details on this component are in Chapter 4, which presents the design and implementation of user preference mining prototype.

In this chapter we presented our model to the research challenges raised in this dissertation. We developed a user preference model that has an intuitive preference measure and strict partial order preference representation. We then discussed how this preference model can be used for mining and representing context-based user preferences in an m-Services environment. To support integration of the mined preferences into an m-services operational environment, we presented a user model for an m-services environment that supports cross personalisation. The User Preference Centred Architecture for Mobile Services (UPCAMS) is presented to put together all the concepts modelled in this chapter.

# CHAPTER FOUR

## PROTOTYPE DESIGN AND IMPLEMENTATION

### 4.0 Introduction

The previous chapter presented the User Preference Centred Architecture for Mobile Services (UPCAMS), an architecture for mining and usage of context-based user preferences in the access and provision of m-services. In this research work our main focus is on the User Preference Mining component of UPCAMS, the User Preference Miner, and its peripheral components. Hence, in this chapter we present the design (Section 4.1), implementation (Section 4.2) and evaluation (Section 4.3) of the prototype of our user preference mining framework.

### 4.1 Design of a Prototype

This section presents the UML modelling of the User Preference Miner developed in this work and the Design of the peripheral components tightly integrated to it.

### 4.1.1 The Preference Miner in UML

Figure 4.1 shows the use cases for the prototype for user preference mining framework implemented in this research work. The system has four (4) actors and these are: the User Session Manager (USM), the User Session Data Warehouse (USDW), the User Preference Profile (UPP), and the Preference Repository (PR).

The preference miner has one *use case* which is a request for user preferences which is forwarded to the Preference Miner by the User Session Manager. This *use case* has two sub-tasks which are: (1) *query for user item preferences*; which queries for the user's preferences on items from the Preference Repository for initial recommendation to the user, and (2) *mine user item preferences*; As the user interacts with the system the user session data gets updated. The quality of the recommendations can be improved by using the updated user session data in the recommendation process. This is done through the invocation of the *mine for user item preference* task. The *mine for user preference* task has two alternative sub tasks. If a given application has got enough user session data for mining user preferences for a given user in the current context under the current domain the *mine Attribute Value preferences* task is invoked otherwise the application queries for attribute value preferences from similar applications through the *query for attribute value preferences* task. The *mine attribute value preferences* task uses the data from the User Session Warehouse through the

supply data to be mined task. The supply data to be mined task have two subtasks, which are get item details and get user session data.

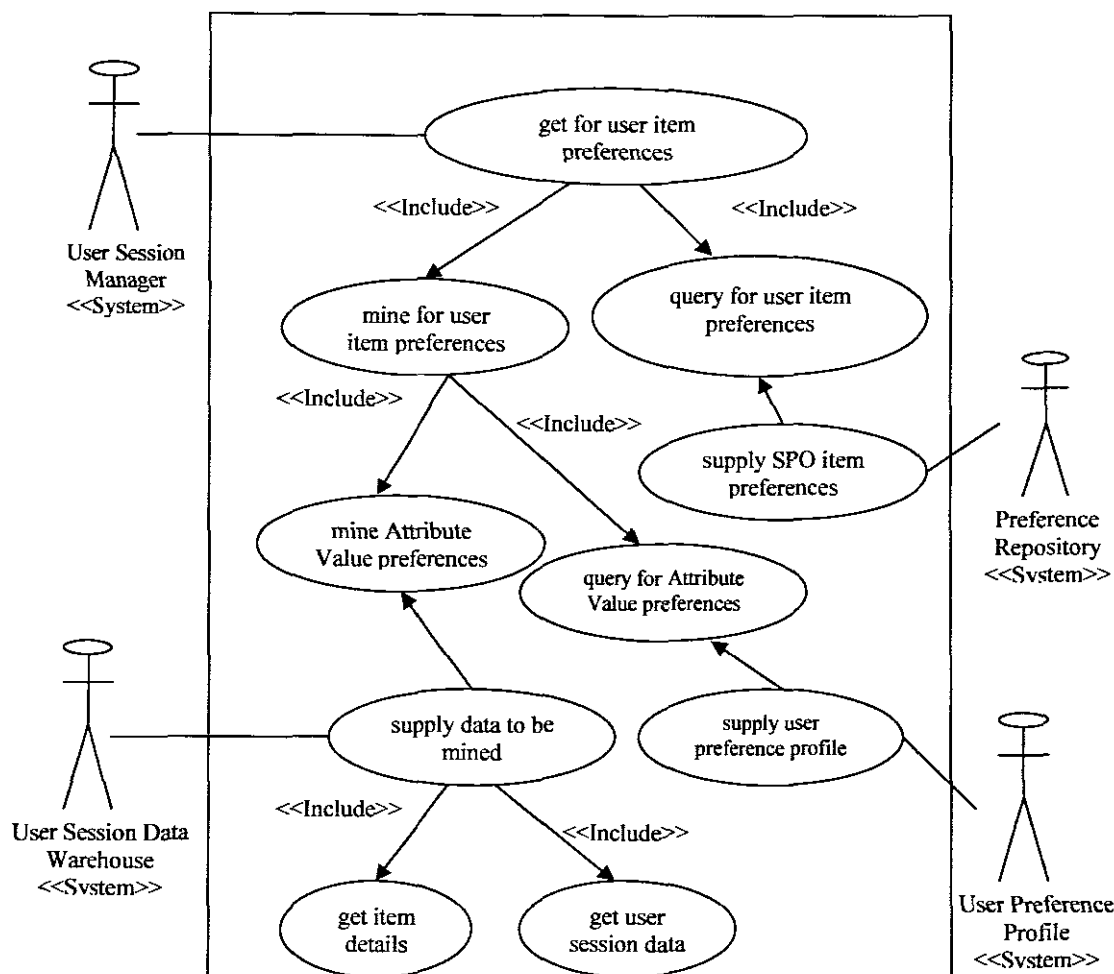


Figure 4. 1: Prototype Use Case Diagram

**A. User Session Manager**

In the Application Layer, the application server is responsible for user session management. It is responsible for generating context based queries for user preferences. At the middleware layer, the user session manager, in the personalisation module, is responsible for session management, which includes

generating context based user preference queries and supplying user session data to the User Session Data Warehouse (See Section 3.6.2).

#### **B. User Session Data Warehouse**

The User Session Data Warehouse is a structured data repository where all the data needed for mining the user preferences is kept. This component is the platform upon which the user preferences are mined. In this case the warehouse can be the application or the middleware User Session data warehouse.

#### **C. User Preference Profile**

This is an XML data repository that stores the mined attribute value preferences of each and every user, who have interacted with the application/system. This repository can also be queried for user attribute value preferences for mining user preferences in similar applications.

#### **D. Preference Repository**

This is an XML data repository that stores the mined Strict Partial Order categorical preferences of each and every user, who have interacted with the application/system. This repository is also queried for item SPO preferences within the same application at the beginning of every session for a returning user.

The activity diagram in Figure 4.2 shows the dynamic nature of our user preference miner prototype. The structural view of the preference miner is shown

in the class diagram in Appendix C. The preference mining process starts with the user preference miner requesting for data for mining user preferences from the User Session Data Warehouse. If the user session data warehouse has enough data for mining user preferences the algorithm for mining attribute value preferences is invoked. The results from this algorithm are used by the matching algorithm to compute item preferences and are also sent to the User Preference Profile to update the profile. The item preferences are then input into the algorithm for mining Strict Partial Order categorical preferences, which gives SPO categorical preferences as output. The mined SPO categorical preferences are used to update the Preference Repository. If the maximal set of the SPO categorical preferences is greater than the maximum number ( $n$ ) of Items that can be recommended to a given user, the maximal set is passed to the algorithm for mining EXPLICIT preference, otherwise the maximal set is recommended. The algorithm for mining EXPLICIT preferences represents maximal set in a Directed Acyclic Graph (DAG). If the DAG's maximal set is still more than the maximum number ( $n$ ) of items that can be recommended to the user, the top  $n$  items in the DAG's maximal set are recommended, otherwise the maximal set is recommended.

#### 4.1.2 User Session Warehouse Data Model

The User Session Data Warehouse model defines a general structure for handling user session and item details data to provide a platform for the user preference mining framework designed in this work. The warehouse was sits on

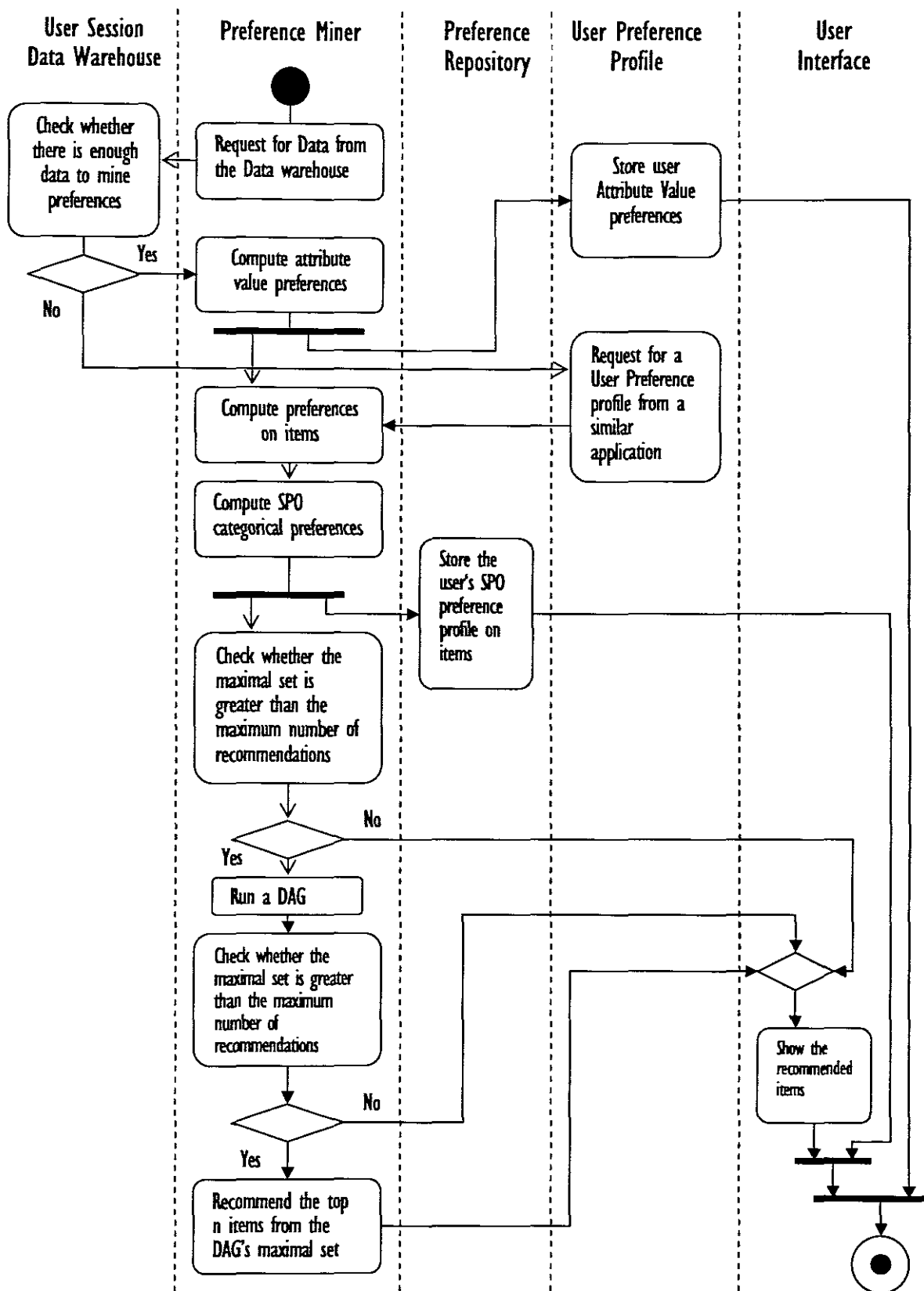


Figure 4. 2: User Preference Miner Activity Diagram.

two databases. The first one is kind of a transaction log that holds all the user session data. The second one is a persistent database that holds all the item details. In the application layer the user session database gets its session data from the application server and in the middleware the session data is derived from the user session manager. The Service registry acts as the item details database in the middleware layer.

Dimensional modelling is used to model the warehouse. Dimensional modelling is a logical design technique that seeks to present the data in the standard, intuitive framework that allows for high access performance. The snowflake design is chosen as the preferred dimensional modelling technique over the star schema because of the following reasons:

- the query performance is improved due to minimised disk storage for data, and
- the general performance is improved by joining smaller normalised tables rather than denormalised tables in the star schema.

The data warehouse model is shown in Figure 4.3. In order to construct the data warehouse model, the following design principles were considered:

- Data retrieval of UserId, DomainID, CID, and item attributes must be optimised, and
- The warehouse should support retrieval of items at different levels of categorisation (e.g. domain, sub-domain, category and sub-category,

e.t.c). To this effect we used domain modelling framework in designing the item details database.

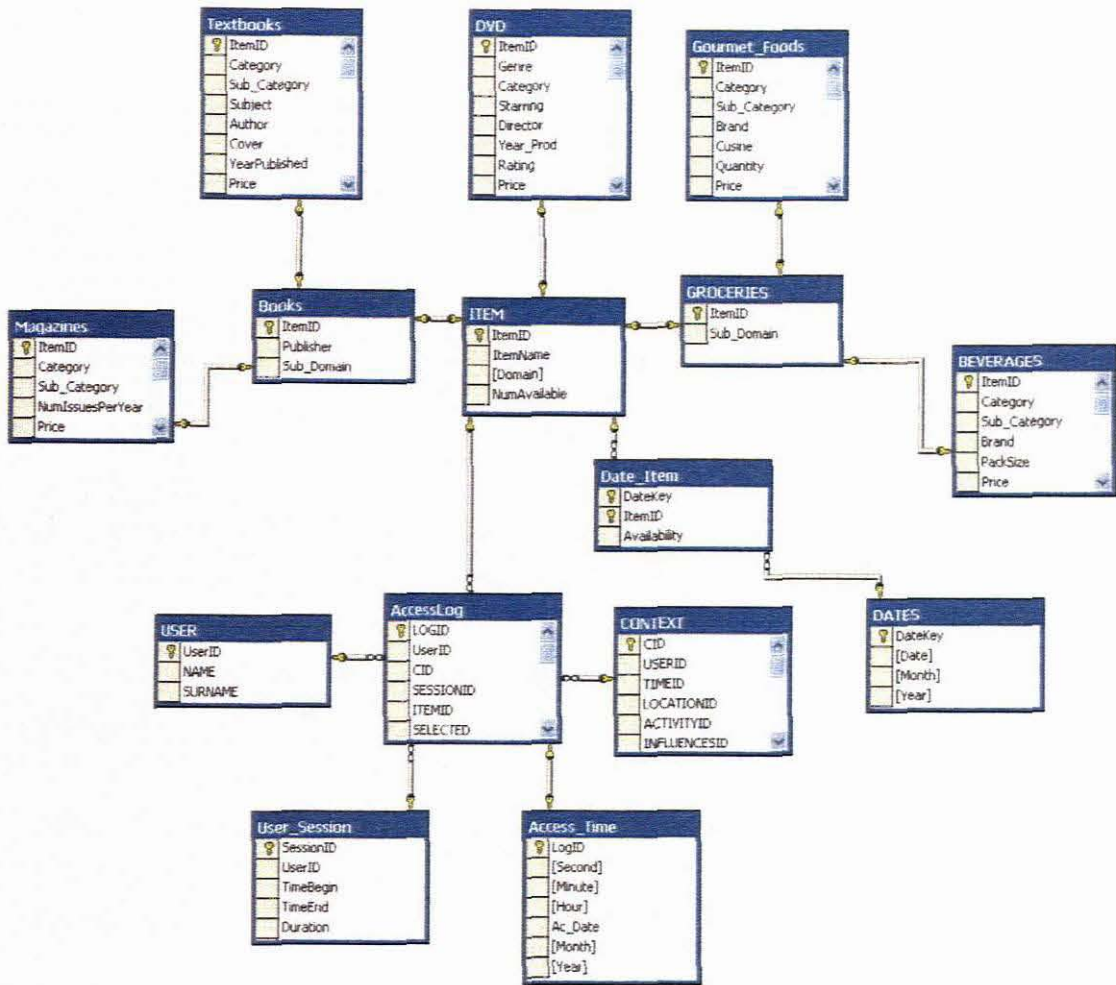


Figure 4. 3: The ERD for the data warehouse model

### 4.1.3 Context-Based User Preference Profiles

With the increasing popularity of Service Oriented Architectures and the enabling technologies such as web services, some intermediate results of the preference

model developed in this work can be shared among similar applications. This is despite the seemingly obvious security, custody and privacy issues, which still need to be solved.

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserProfile SYSTEM "UserProfile.dtd"[ ]>
<!--
a repertoire of preferences that may occur in the DVDs application
-->
<UserProfile>
  <UserIdentifier Value="900000001">
    <UserProfileData Value="DVDs">
      <Context Value="10000001">
        <TimeStamp dateTime="2007-05-27T04:38:18"/>
        <Attribute Value="Genre">
          <AttributeValue Key="Action" Value="2.89"/>
          <AttributeValue Key="Comedy" Value="2.56"/>
          <AttributeValue Key="Adventure" Value="0.098"/>
          <AttributeValue Key="Classical" Value="-1.67"/>
        </Attribute>
        <Attribute Value="Actor">
          <AttributeValue Key="Mel Gibson" Value="1.67"/>
          <AttributeValue Key="Sylvester Stalone" Value="-3.89"/>
          <AttributeValue Key="Sharon Stone" Value="0.9"/>
          .
          .
          .
          <AttributeValue Key="Tom Cruise" Value = "-0.67"
        </Attribute>
      </Context>
      .
      .
    </UserProfileData>
    .
    .
  </UserIdentifier>
  .
  .
</UserProfile>

```

Figure 4. 4: User Preference Profile

Figure 4.4 shows a sample of an XML repository for some intermediate results from our user preference mining framework. We named this repository *User Preference Profile*. The DTD for the User Preference Profile is shown in Appendix A. The User Preference Profile holds each registered user’s

preferences on the features/attribute values for a given product domain under a given context. The Attribute Value preferences will then be used to compute the preferences on individual items using equation 3.4 presented in Section 3.2.2. As long as the applications involved follow a standard way of defining items the User Preference Profile generated from one application can be used in another similar application. The need to allow similar applications to share User Preferences Profiles makes XML the best option to implement the User Preferences profiles.

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE PreferenceRepository SYSTEM "preference.dtd"[]>
<!--
  a repertoire of preferences in an m-commerce application
-->
<PreferenceRepository>
  <UserIdentifier Value = "900000001" >
    <PreferenceData Value ="DVDs">
      <Context Value = "100000001">
        <TimeStamp dateTime="2007-05-27T04:38:22" />
        <Preference>
          <POS att="DVDs">
            <POSSet>
              <Value val="Lethal Weapon" />
              <Value val="The Matrix" />
              <Value val="Season Five" />
              <Value val="The Blues Brothers" />
              <Value val="Moonstruck" />
              <Value val="Nine Months" />
              .
              .
              .
              <Value val="Winners and Sinners" />
            </POSSet>
          </POS>
        </Preference>
      </Context>
      <Context Value = "100000036">
        .
        .
        .
      </Context>
    </PreferenceData>
    .
    .
    .
  </UserIdentifier>
  <UserIdentifier Value = "9000000002">
    .
    .
    .
  </PreferenceRepository>

```

Figure 4. 5: User Preference Repository

#### 4.1.4 Context-Based User Preference Repository

After preferences on items have been computed, what remains is the intelligent storage and usage of such preferences. The preference repository will have to give an appropriate storage structure of the preference model developed in this work. XML is also used to implement the preference repository. Figure 4.5 show a sample XML file for the preference repository. The DTD for the Preference Repository is given Appendix B.

### 4.2 Implementation Environment

The Preference miner was implemented using the JBuilder 2005 IDE (Integrated development Environment) for Java with Java Development Kit Version 6 (JDK6). The underlying database used is the MS SQL server (using default settings). The preference miner and the SQL server were running on the same machine with a 2.8 GHz processor and 512 megabytes memory CPU.

The implementation of the User Preference Miner prototype necessitates a test environment for testing the functionalities of the preference prototype. The Interface for our user preference miner is shown in Figure 4.6. First the userID is input in the upper area. A click of the 'Enter' button in the upper area searches for the contexts under which a given user has interacted with the application before. Selection of the subsequent parameters in the middle area and a click of the 'Execute' button give the results in the lower area.

**User Preferences Miner**

Enter the following input to the system

User ID  Enter

User ID  Max No of Preference Categories

Context  Strong Negative Preferences Threshold

Domain  Include Unavailable Items

Max No of Recommendations  Silhouette Threshold

Execute

**Recommended Items**

The Quest for Global Dominance: Transforming Global Presence into Global Competitive Advantage  
 Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications  
 Guide to Econometrics  
 Auditing and Assurance Services

**Preference Categories**

EXPLICIT\_POSEXPLICIT\_SOFT\_NEG

POS CATEGORY [1]

30004 Auditing and Assurance Services 0.6208038680048823  
 30037 Global Business Today 0.5761015041853498  
 30005 Auditing & Assurance Services: A Systematic Approach 0.6133389636671795  
 30041 The Quest for Global Dominance: Transforming Global Presence into Global Competitive Advantage 0.6722230357215095  
 30008 The Internal Auditing Pocket Guide 0.6169574106715381

Figure 4. 6: The User Preference Miner Test environment

We used the GUI shown in Figure 4.7 for performance evaluation of our preference mining algorithms. The procedure to get the results in the lower layer of Figure 4.7 is the same as the one for Figure 4.6. In the lower layer of Figure 4.7 the execution time up to the end of a given algorithm is recorded. The execution time for a given algorithm was derived by subtracting execution time up to the end of the same algorithm from that of the one preceding it.

**User Preferences Miner Evaluation**

Enter the following input to the system

|                           |                      |                                           |
|---------------------------|----------------------|-------------------------------------------|
| User ID                   | <input type="text"/> | Enter                                     |
| User ID                   | 90000031             | Max No of Preference Categories: 10       |
| Context                   | 10000136             | Strong Negative Preferences Threshold: 10 |
| Domain                    | DVDs                 | Include Unavailable Items: YES            |
| Max No of Recommendations | 4                    | Silhouette Threshold: 0.51                |
| Execute                   |                      |                                           |

| Execution Time in Mill-seconds      |      | Decision Parameters               |         |
|-------------------------------------|------|-----------------------------------|---------|
| Data Access                         | 969  | POS Silhouette                    | 0.57711 |
| Up to Matching features to products | 1250 | SOFT_JIEG Silhouette              | 0.44348 |
| Upto K-Means algorithms             | 7109 | Number of Items in First Category | 24      |
| Upto the DAG                        | 7625 | Number of Items Clustered         | 103     |

Figure 4. 7: User Preference Miner Performance evaluation environment

### 4.3 Evaluation of the Preference Mining Framework

The evaluation of the preference mining framework developed in this work is twofold: First, in Section 4.3.1, the effectiveness of the user preference model is evaluated using some simulated users. Second, the performance of the algorithms developed in this work to form components of the preference miner is evaluated in Section 4.3.2. The data for this evaluation was generated by simulating user sessions. Literature (e.g. Kobsa and Fink, 2003, Montgomery and Faloutsos, 2000) has shown that *url* hits and viewings tend to follow a Zipf density function. To this end, the simulation of users we carried out in this work used a Zipf density functions to generate user transactional data.

### 4.3.1 Effectiveness of the User Preference Model

To evaluate the accuracy of our user preference mining framework we have chosen to prove that our user preference model reflects the user preference accurately. This section presents the Test design of the experimental environment for evaluating the effectiveness of the preference model presented in this work and the test results.

#### 4.3.1.1 Test Design

Preference profiles of 35 users were generated through passing log file data (all hits) generated in Matlab 7.1 through the preference miner prototype developed in this work. Two Zipf density function were considered in generating the user profiles and these: the one for item selection (viewings) and the other for all hits.

Item selections (viewings) were generated from a Zipf Probability Density Function (pdf) derived from the MATLAB 7.1's Generalised Pareto Probability Density Function (gppdf) with shape, scale and threshold parameter of 1.25, 1, and 0 respectively. The shape parameter for item selection was chosen on the basis of the observation made in (Montgomery and Faloutsos, 2000) that the maximum likelihood estimate for the Zipf shape parameter of url viewings (selections) and not hits (analogical to recommendations in this work) is 1.25. The findings by (Montgomery and Faloutsos, 2000) also showed that even though the number of sessions per user and the number of viewings per month were

growing exponentially, the number of viewings per session is stable through time at an average of 48 viewings per session. Based on this fact and the fact that the user preference mining framework developed in this work is to be used in a mobile computing environment, where the sessions are bound to be smaller and hence the number of viewings, we opted for viewings ranging from 1 to 25 viewings per session, with a few outlier values above 25. The average number of views per session in the data we simulated was eleven (11).

The log data (all hits) was generated using a Zipf density function derived from Matlab 7.1's Generalised Pareto Probability Density Function (gppdf) with a shape parameter 1.4, scale parameter 1 and threshold parameter 0. The shape parameter of the Zipf pdf for generating log data was chosen taking special care that the number of options available for a user to select is always greater than the number of selections made for each item. Each session had an average of 18 hits and 11 viewings.

The resulting user profiles created from the process discussed above is then used to query the product database. A total of 51990 log relations were generated for all the users. The synthesised log data was then passed through the preference miner to investigate whether the preference miner will be able to return the original user profiles as illustrated in Figure 4.8. A comparison of the item preferences detected with the original item preferences is used to show the effectiveness of the developed preference mining framework.

The following parameters (as defined in (Holland et al, 2003)) are used to evaluate the accuracy of the preference model presented in this work:

$$\text{Precision} = \frac{\text{number of correctly detected item preferences of user } i}{\text{number of all detected item preferences of user } i} \quad 4.1$$

$$\text{Recall} = \frac{\text{number of correctly detected item preferences of user } i}{\text{number of all item preferences of user } i} \quad 4.2$$

Preference Recall measures how good the model developed in this work is, in making sure that there are no missing relevant recommendations. Preference Precision measures how good the framework is in reducing irrelevant recommendations. A good preference model is expected to optimise these two parameters.

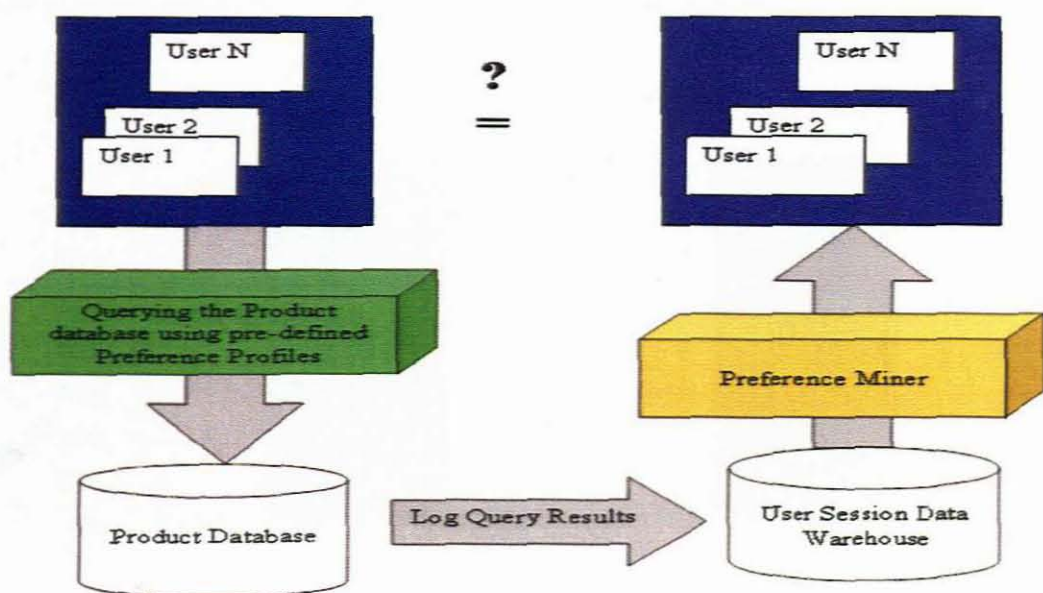


Figure 4. 8: Experimental set up for evaluating the preference mining framework

### 4.3.1.2 Test Results

All the sets of results presented in this section were obtained from the preference miner with a silhouette value of at least 0.51 for the k-means algorithm. If the silhouette is less than 0.51, it is concluded that there is no reasonable structure found in the data and hence the clustering algorithm returns the unclustered data set. The silhouette value was chosen based on the partitioning suggested in (Holland, 2003). Using a minimum silhouette value of 0.71 made the clustering algorithm too restrictive, so much that it was only recognising the lower level preferences (i.e POS, SOFTNEG, and POSSOFTNEG).

In domains where the Closed World Assumption (CWA) holds, the model was found to be significantly accurate in predicting user preferences with preference recall and precision values of about 82% (see Figure 4.9).

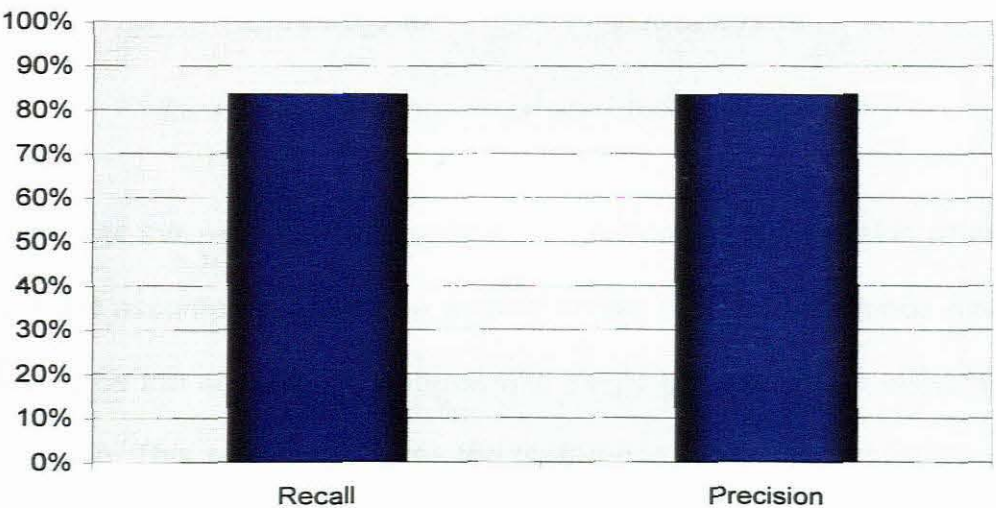


Figure 4. 9: Preference Recall and Precision (CWA)

Figure 4.10 shows the results obtained from running the preference miner in domains where the Closed World Assumption does not hold. The first set of results was obtained from running preference miner on the generated log data allowing maximum of 10 preference categories for each run of the k-means algorithm and the second was run allowing 20 preference categories for each run of the k-means algorithm. A tighter clustering reduced the preference recall and increased the preference precision slightly.

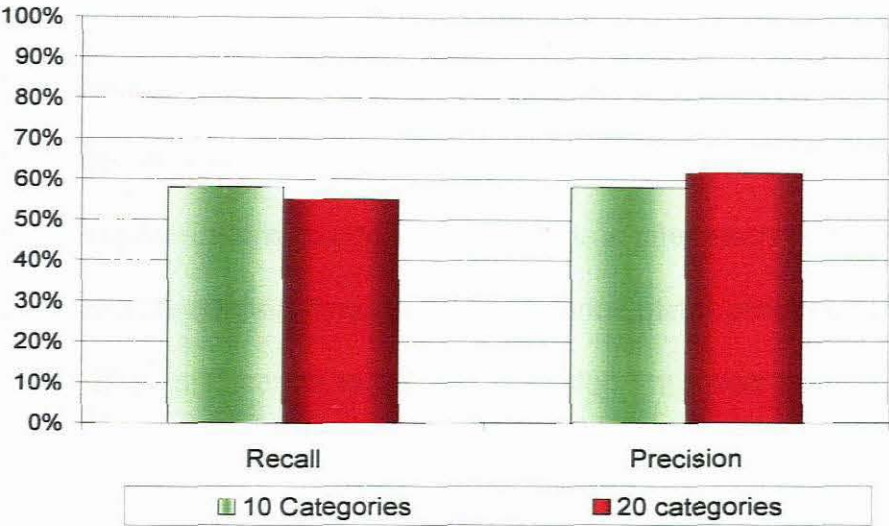


Figure 4. 10: Preference Recall and Precision ( $\neg CWA$ )

An analysis of the nature of our preference measure in the domains where the closed world assumption cannot be applied shows that the preference measure is bound to be too sensitive to features with single hits and single viewings in a given domain. This sensitivity makes the preference model very anticipative and this may result in incorrect recommendations. The features concerned will load high in all the items they appear in thereby giving them high preference values. Over a period of time the model will always work to correct this anomaly through

the fact that persistent recommendation of a feature which is never selected reduces its preference value and hence its load on the items that contains it.

One way to counter this anomaly is to have as much historical data as possible to dump down the anticipative effect of our model. Hence in our experimental set up, we combined the original data which was used to generate initial user profiles (training data set) and the data obtained from querying the product database using the generated user profiles (Test data) to create a single data set. In reality the mined user preferences get sharper after every session, and this could not be captured in the experimental set up used in this research work due to the fact that, it will be cumbersome to update the Test data after each and every session. A run of the combined data through the preference miner produced the results in Figure 4.11. Both preference recall and precision improved from about 60% to about 70%.

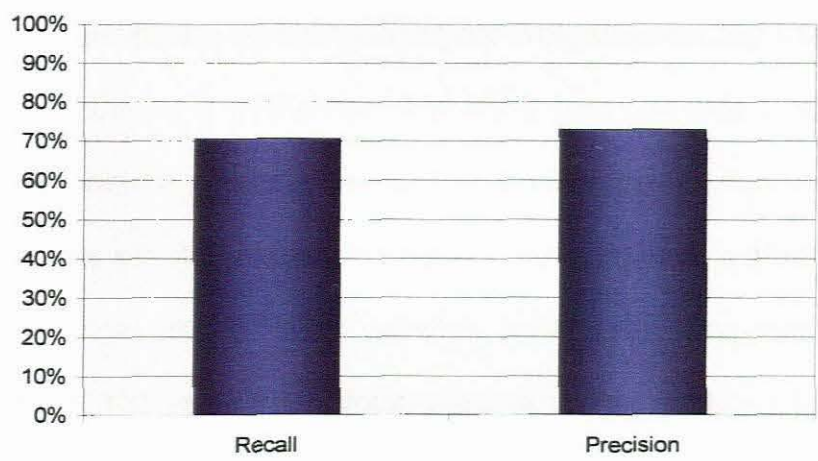


Figure 4. 11: Preference Recall and Precision ( $\neg CWA$ , Combined Data)

### 4.3.2 Quality of the User Preference Mining Algorithms

This section evaluates the quality of the user preference mining algorithms developed in this work in terms of capacity scalability. Apart from accessing data in the data warehouse, the developed preferences mining framework employs three algorithms which are susceptible to scalability problems. These algorithms are:

- the algorithm to match the feature preferences to the product profile,
- the clustering Algorithm, and
- the Directed Acyclic Graph algorithm.

In the next two sub-sections we discuss the test design and results for evaluation of the quality of the aforementioned algorithms.

#### 4.3.2.1 Test Design

The data for evaluating the performance of our preference mining framework was generated by simulating a single user with 3000 sessions with a total of 50000 hits under five different contexts. Thus each session on average had about 17 hits. The data was simulated using MATLAB 7.1's gppdf with a shape parameter of 1.4 for hits (logs) and 1.25 for viewings (selections). Log files of sizes of 10000, 20000, 30000 and 40000 tuples were extracted randomly from the data set mentioned above. The individual log files including the original 50000 tuple file were passed through the preference miner and the time taken by each algorithm to do its task were observed.

### 4.3.2.2 Test Results

Figure 4.12 shows that generally, the preference miner does not scale well with the increase in the number of tuples. Against this background, we investigated each algorithm's effect on the overall execution time as the number of tuples in the user session data warehouse increases.

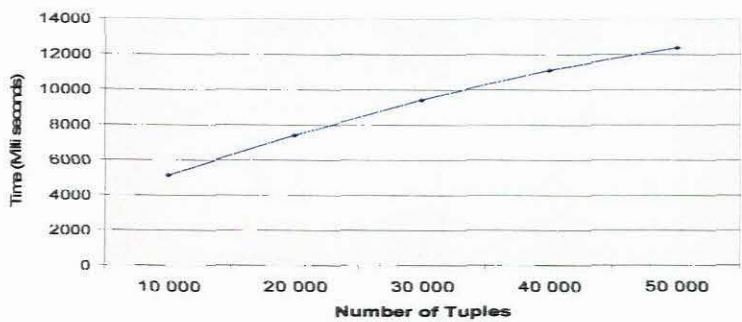


Figure 4. 12: Scalability of our preference mining with increases in the number of tuples

Generally, regardless of the JDBC used, data access does not scale well with the increase in the number of tuples. Figure 4.13 shows that the time required to access the data from the database increases with the increase in the number of tuples per user.

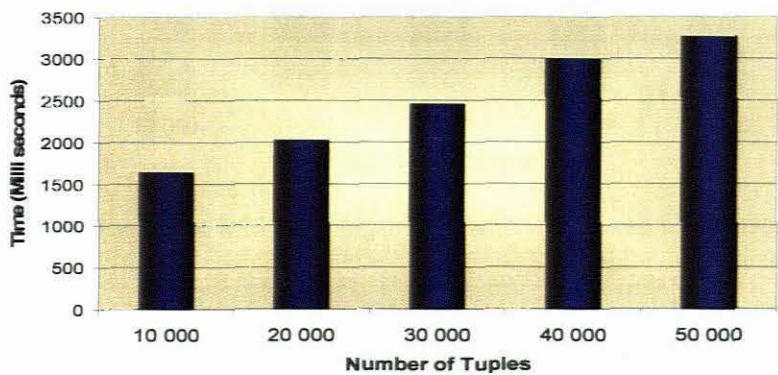


Figure 4. 13: Scalability of data access from the data warehouse with increases in the number of Tuples.

The algorithms to match feature preferences to product preferences and the k-means clustering algorithms are not affected by the increase on the number of tuples (see Figure 4.14 and Figure 4.15). This is because these algorithms only deal with the summarised output from the data warehouse.

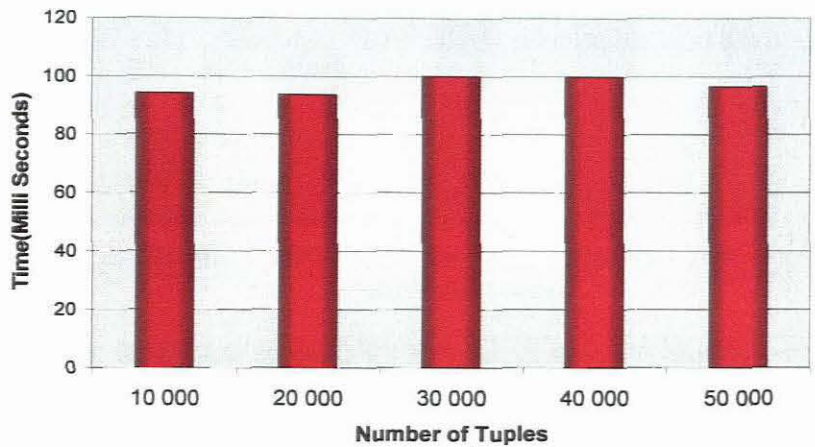


Figure 4. 14: Scalability of the matching algorithm with increases in the number of tuples.

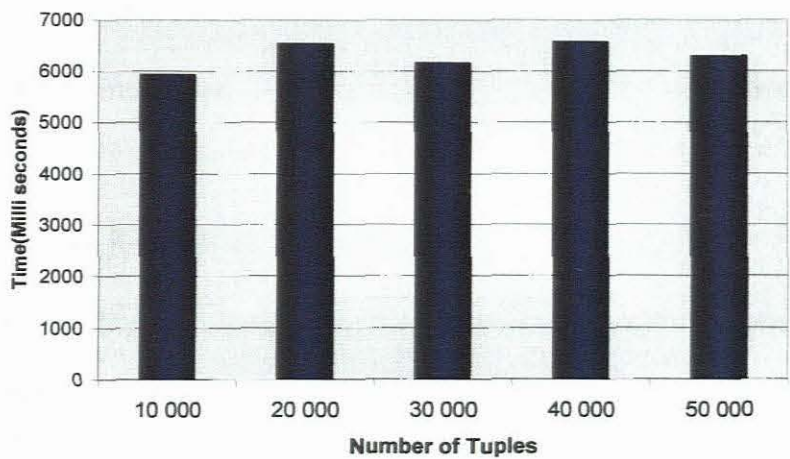


Figure 4. 15: Scalability of the k-Means algorithm with increases in the number of tuples

Figure 4.16 shows that the construction of a Directed Acyclic Graph (DAG) to represent relative preferences on items in preference category is affected by the

number of tuples. The time to synthesise a DAG increases with increases in the number of tuples per session.

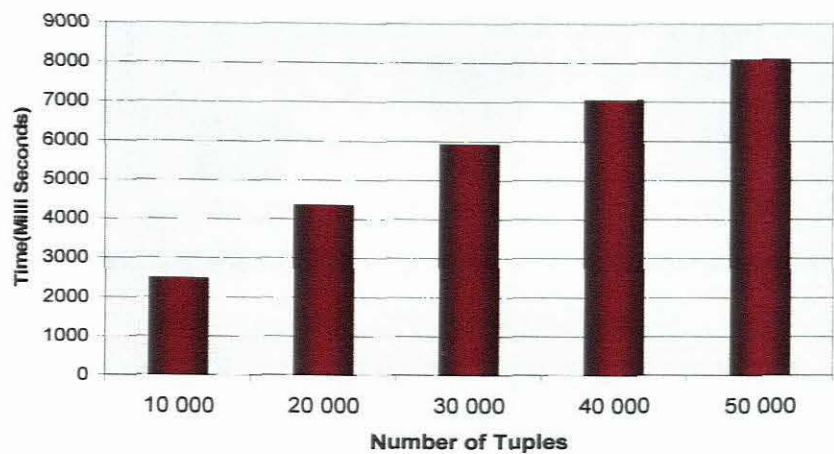


Figure 4. 16: Scalability of the DAG with increase in the number of tuples

An analysis of the results presented in Figure 4.13 to Figure 4.16 reflects that database access and the DAG algorithm do not scale well with the increase in the number of tuples in the database. The matching algorithm and the k-means are not affected by increases in the number of tuples in the database.

Scalability of the k-means: other factors

The major scalability constraint on the k-means algorithm derives from the increase in the number of clusters the algorithm should run in-order to get a relatively optimal clustering structure. Figure 4.17 shows that the time to execute the k-means algorithm increases exponentially with the increase in the number of preference categories (clusters) to be synthesised.

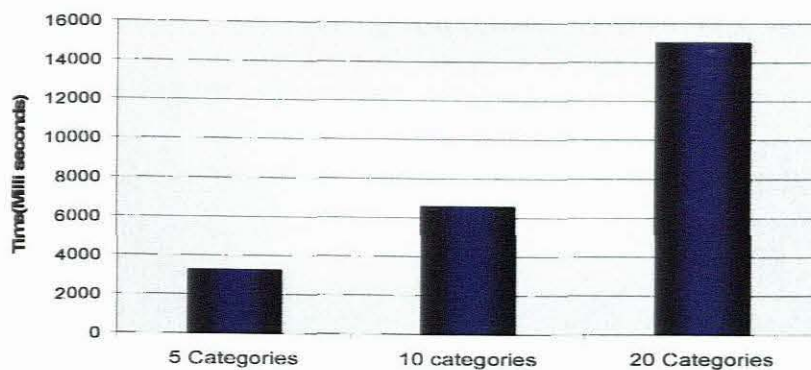


Figure 4. 17: Scalability of the k-means algorithm with increase the number of clusters

Apart from this, the execution time of the k-means is also adversely affected by the increase in the number of products (Figure 4.18) in a given domain.

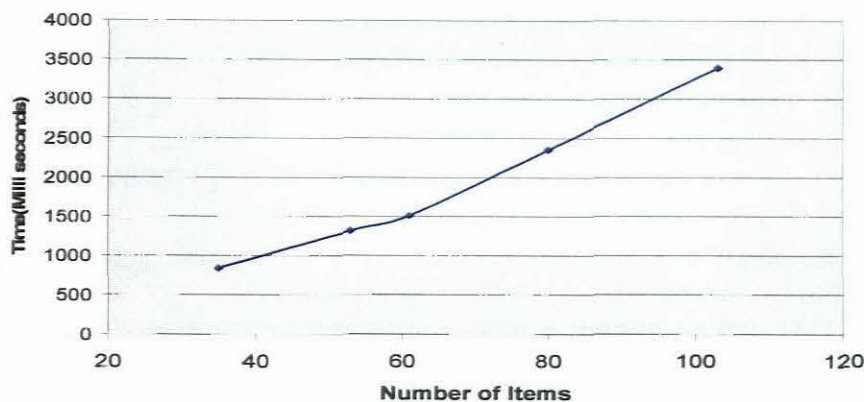


Figure 4. 18: Scalability of the k-means algorithm with increase in the number of items to be clustered

### Scalability of the DAG: Other factors

The DAG, like the k-means is negatively affected by the increase in the number of items whose relative preferences should be synthesised. The DAG performs (see Figure 4.19) well when the k-means have found well defined cluster

structure. In this case DAG has to represent considerably few items and therefore its execution time will be less.

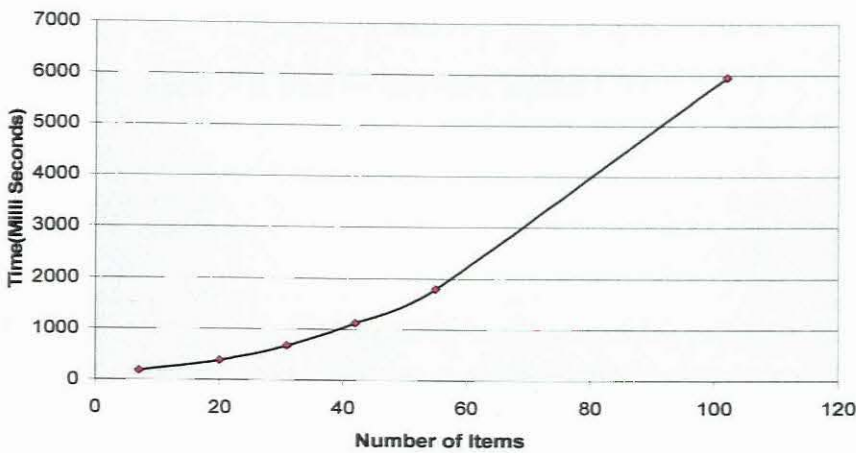


Figure 4.19: Scalability of the DAG algorithm with increases in the number of items to be represented.

### Trade- off between Clustering and DAG algorithm

From the foregoing discussion it can be deduced that a trade-off need to be reached on which of the two algorithms, the *k-means* or the *DAG*, should be relieved at the expense of the other while still maintaining the effectiveness of our preference mining framework. Running the user preference miner with the clustering algorithm returning a single cluster, shift the whole load of preference extraction to the DAG algorithm and hence the DAG will take more time to execute. A run of the preference miner with the clustering algorithm returning a number clusters increases the time it takes for the clustering algorithm to execute, but it lessens the number of items the DAG has to represent and hence its execution time will be less. Figure 4.20 shows that as the number of tuples per user increases, it is profitable to relieve the DAG at the expense of the clustering

algorithm. This is because the clustering algorithm is not affected by the number of tuples in the user session. Thus, provided that the clustering algorithm finds a well defined structure our user preference mining framework scales well with increases in the number of tuples in the data warehouse.

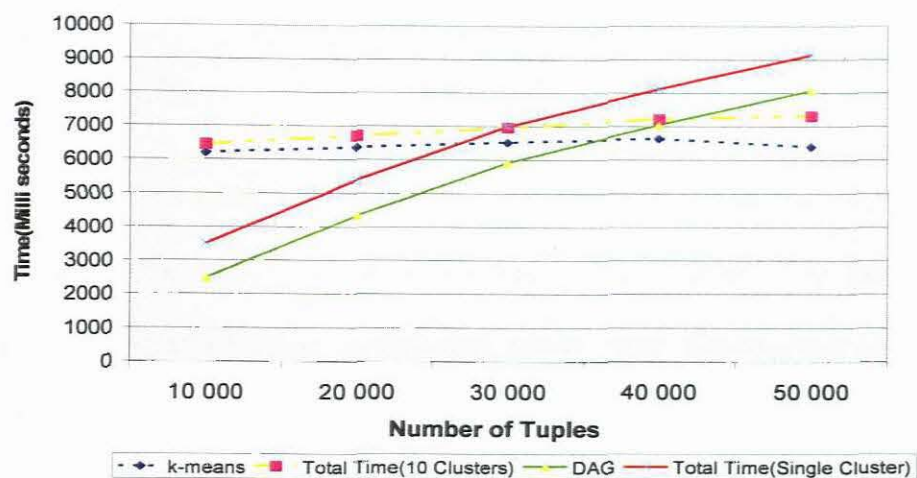


Figure 4. 20: Effect of the number of clusters on the total execution time as the number of tuples increases.

In this Chapter, we presented the implementation of our user preference mining framework prototype. The prototype was used to evaluate our user preference mining framework with respect to the effectiveness of our preference model and the quality of the preference mining algorithms.

# CHAPTER FIVE

## CONCLUSIONS AND FUTURE WORK

### 5.0 Introduction

This research work's goal was to develop a user preference model and mining framework that integrates into an m-services environment. To this end, we developed a user preference model that gives user expressiveness and an intuitive interpretation of a preference. To address the dynamic nature of user preferences in a mobile computing environment, a context model for an m-services environment was given. We also developed algorithms suitable for mining context-based user preferences in a mobile computing environment based on the preference model and the context model we developed in this work. The preference miner and all its peripheral components were put together in the form of a Framework that integrates into an m-Services environment. The framework is meant to support advanced personalisation in an m-services environment by allowing sharing of user profiles among similar applications. To ensure this sharing, a user model that supports cross personalisation has been defined. A prototype of the preference miner and its peripheral components (the user session data warehouse, User Preference Profile repository, preference repository) were used to evaluate our user preference model and quality of the preference mining algorithms.

The remainder of this chapter is organised as follows: Section 5.1 presents a discussion of the results obtained in Section 4.3.1 and Section 4.3.2, and their implication on our model. Section 5.2 presents our future research directions.

## 5.1 Conclusions

The expected impact of our preference mining framework is two fold:

- Our user preference model demonstrates both an intuitive measure of a preference and user expressiveness in the representation of preferences for automatically analysing preference models, which none of the existing memory-based preference model can do. Most of these preference models use indirect measures of a preference and they do not handle the issue of preference representation at all, and
- Our framework shows the feasibility of mining of context-based user preferences in an m-services environment.

An evaluation of the effectiveness of our user preference model showed that the model is very promising. Performance analysis of the algorithms we developed in this work, presented in (Jembere et al, 2007), reflected that our user preference mining framework scales relatively well with increase in the number of tuples to be spanned through, provided that much of the preference extraction process is not shifted from the clustering algorithm to the DAG algorithm. An attempt to increase the maximum number of permissible clusters to reduce the time it takes

a DAG to synthesis Strict Partial preferences, within a preference category, resulted in the ballooning of the overall execution time of the preference miner. Thus, for optimal performance of the preference miner, the number of permissible preferences must be kept as minimal as possible.

Having evaluated the preference mining framework developed in this work, it is also important to benchmark it with modern recommendation tools. The experimental comparative analysis to this effect is beyond the scope of this work. However, it is important to analyse how it compares to them from a logical point of view.

Classical recommender systems use historical data on user preferences to predict items the user might be interested in. Approaches to recommender systems can broadly be categorised as memory based, if they operate over the entire data to make predictions and as model based if they use the historical data to build a model which will then be used for predictions (Zhang et al, 2002). The user preference model presented in this work is one of the memory based approaches to item recommendations.

Apart from the lack of intuitive preference measures and user expressiveness, most of the approaches to memory-based recommender systems do not scale with increases in the number of tuples in the underlying database, since they have to run on multidimensional data. Our preference mining framework first

reduces the multi-dimensional structure to a single dimension, without losing any information, before applying computationally expensive algorithms. Thus the preference model developed in this research work can be logically viewed as a content-based vector similarity model enhanced to give an intuitive measure of a preference, an intuitive representation of preferences, and to provide computational efficiency.

One other significant problem in memory-based preference model is the latency problem - i.e. the problem of how the system should behave when they have low volumes of historical data (Sicilia and Garcia, 2004). Though data sparseness results in poor predictions, it is not critical to our preference model. Even in instances where the user has just a few tuples of data and a single item selection, our framework, still allows detection of Strict Partial Order categorical preferences.

A comparison of the results on the effectiveness of the model in the domains where the CWA holds and where it does not hold, shows that the model is more effective in domains where the CWA holds than in those domains where the CWA does not hold. This might be, because of the fact that our preference model is more anticipative in domains in domains where the CWA does not hold. For instance, if a feature only appears in one hit and it was selected, the feature will have the maximum possible loading on the preference values of all the items it appears in. This results in these items having high preference values and may

subsequently be in the maximal set. Thus our model, depending on the loading of the user's other preferences on the product features, anticipates that such items are among the most preferred items and hence it may recommend them. If the features in question are not significantly preferred, the preference model will eventually correct this with time as the items will keep getting recommended and yet never selected, thereby reducing the features preference to approach its true value. Thus our model is bound to be robust in detecting the user's new preferences and the subsequent correction of some biased preferences. However this still needs to be investigated using real life data.

Apart from the above discussed merits of our preference mining framework, our results showed that our framework does not scale well with increases in the number of items in the domain whose preferences are to be synthesised. The negative effect of the increases in the number of items on the performance of our preference mining framework can be kept minimal by ensuring that the maximum number of preference categories to be detected by the clustering algorithm is kept small enough not to shift the bulk of the work to the DAG.

Another potential point of weakness of our preference mining framework emanates from the fact that our framework assumes that the user's preferences on a set of items in a given domain are determined by the user's preferences on the domain's common attributes. In reality some items have some extra features, not possessed by other items in the same domain. It is possible that the user's

preferences on a given set of items might be highly influenced by these extra features and hence our preference model will not be able capture this.

## 5.2 Future Work.

This study raised a lot of issues which need further study. Our context model was only analysed from a theoretical point of view. The envisaged next step is to investigate this model's performance with simulated data. The infrastructure for user preference mining and use in mobile computing shows that preference mining and usage can be done in two layers, that is the middleware and the application layer. Our evaluation only investigated the use of the model in the application layer. Its evaluation in the middleware layer is still to be done.

# BIBLIOGRAPHY

Adomavicius, G., Tuzhlin, A. (2002). Expert-Driven Validation of Rule-Based User Models for Personalisation Applications, *Data Mining and Knowledge Discovery*, 5

Alvin, T.S., Chuang, S. N. (2003). MobiPADS: A Reflective Middleware for Context-Aware Mobile Computing, *IEEE Transactions on Software Engineering*; December 2003 (Vol. 29, No. 12), pp. 1072-1085.

Balke, W. T., Wagner, M. (2003). Towards Personalised Selection of Web Services; *In Proceedings WWW, 2003*, Budapest, Hungary, May 2003

Barkhuus, L. (2005). The Context Gap: An Essential Challenge to Context-Aware Computing; *Ph.D. dissertation, The IT University of Copenhagen*, 20 January 2005 [http://www.itu.dk/people/barkhuus/lou\\_thesis05.pdf](http://www.itu.dk/people/barkhuus/lou_thesis05.pdf) (Last accessed 29 March 2007)

Biegel, G., Cahill V. (2004). A Framework for Developing Mobile, Context-aware Applications, *In proceedings of the 2<sup>nd</sup> IEEE Conference on Pervasive Computing and Communications*, Orlando, FL, March 14-17.

Bonett, M. (2001). Personalisation of Web Services: Opportunities and Challenges. *ARIADINE*, 28, June 2001, <http://www.ariadne.ac.uk/issue28/personalization> (Last accessed on 24 April 2006).

Bormann, F., Flake, S., Tacke, J., Zoth C. (2005). Towards Context-Aware Service Discovery: A Case Study for a new Advice of Charge Service. <http://www.eurasip.org/content/Eusipco/IST05/papers/502.pdf> (Last Accessed on 12 June 2007)

Byun, H. E., Cheverst, K. (2001). Exploiting user models and context-awareness to support personal daily activities; *in Proceedings of the workshop on user modelling for context-aware applications (UM 2001)*, Germany, July 2001.

Coppola, P., Mea, V. D., Gaspero, L. D., Mizzaro, S., Scagnetto I., Selva A., Vassena L., Rizio, P. Z. (2005). MoBe: A Framework for Context-aware Mobile Applications; *Workshop on Context-Aware Proactive Systems*, Helsinki, Finland; 16-17 June 2005.

Dey A. K., Abowd, G.D. (2000). Towards Better Understanding of Context and Context-awareness, *In Workshop on The What, Who, Where, When, and How of Context-Awareness*.

Douikeridis, C., Loutas, N., Vazirgiannis, M. (2006). A System Architecture for Context-Aware Service Discovery; *CWS Preliminary version, Electronic Notes in Theoretical Computer Science*.

Douikeridis, C., Vazirgiannis, M. (2004). Querying and Updating a Context-Aware Service directory in Mobile Environments; *In Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI'04)*.

E-Speak (2001), E-Speak Architectural Specification Release A.0, <http://www.e-speak.hp.com/media/a0/architecturea0.pdf>, (Last accessed on 29 June 2006)

Fahy, P., Clarke, S. (2004). CASS-Middleware for Mobile Context-Aware Applications; *In Mobisys 2004 Workshop on Context Awareness*; [www.sigmobile.org/mobisys/2004/context\\_awareness/papers/cass12f.pdf](http://www.sigmobile.org/mobisys/2004/context_awareness/papers/cass12f.pdf) (Last accessed on 29 June, 2006).

Gehlen, G., Mavromatis, G. (2003). Mobile Web Services Based Middleware for

Context-aware applications; [www.comnets.rwth-aachen.de/436+M5cc1c271ec8.pdf](http://www.comnets.rwth-aachen.de/436+M5cc1c271ec8.pdf) (Last accessed on 29 June, 2006).

Göker, A., Myrhaug, H. I. (2002). User Context and Personalisation. In *Proceedings of the 6<sup>th</sup> European Conference/Workshop on Case-Based Reasoning (ECCBR 2002)*, Aberdeen, Scotland, UK, September 4-7, 2002 <http://www.smartweb.rgu.ac.uk/papers/AGoker.pdf> (Last accessed 29 March 2007)

Gorgoglione, M., Palmisano, C., Tuzhilin, A. (2006). Personalisation in Context: Does Context Matter When Building Personalised Customer Models, In *Proceedings of the sixth International Conference on Data Mining (ICDM'06)*, pp. 222-231

Gu, T., Pung, H. K., Zhang D. Q. (2004). A Middleware for Building Context-Aware Mobile Applications; [www.comp.nus.edu.sg/~gutao/gutao NUS/VTC2004 gutao.PDF](http://www.comp.nus.edu.sg/~gutao/gutao_NUS/VTC2004_gutao.PDF) (Last accessed on 29 June, 2006).

Holland, S. (2003). Preference Mining and Preference Repositories: Design, Algorithms and Personalised Applications, *PhD Thesis, University of Augsburg, German*, 01 January 2003.

Holland, S., Ester, M., Kiessling, W. (2003). A Novel Approach on Mining User Preferences for Personalised Applications. In: *Knowledge Discovery in Databases (PKDD 2003)*, Dubrovnik, Croatia, 22-26 September 2003; pp 204-216 <http://www.cs.uoi.gr/~kstef/PreferenceMining.pdf> (last accessed on 24 April 2006).

Holland, S., Kiessling, W. (2004). Situated Preferences and Preference Repositories for Personalised Database Applications; In *Proceedings of the 23<sup>rd</sup> International Conference on Conceptual Modelling*, [http://www.cs.uoi.gr/~kstef/2004\\_hol\\_kie\\_er2004.pdf](http://www.cs.uoi.gr/~kstef/2004_hol_kie_er2004.pdf) (last accessed on 24 April 2006).

Jameson, A. (2001). Modelling Both the Context and the User; *Personal and Ubiquitous Computing*, Volume 5, Issue 1, Feb 2001, pp 29-33  
[http://www.dfki.de/~jameson/pdf/pete01\\_jameson.pdf](http://www.dfki.de/~jameson/pdf/pete01_jameson.pdf) (last accessed on 24 April 2006)

Jembere, E., Adigun, M. O., Xulu S. S. (2007). Mining Context-based User preferences for Mm-Services Applications: *In Proceedings of the 2007 IEEE/WIC/ACM International Conference on Web Intelligence (WI'2007)*, Silicon Valley, USA, 2-5 November 2007 (Still to be published)

Jembere, E., Adigun, M.O., Xulu, S.S., Emuoyibofarhe O. J. (2006). A Conceptual Model for Supporting Advanced Personalisation in Personalisation in m-Services Applications; *IASTED International Conference on Software Engineering and Applications (SEA 2006)*, Dallas, USA, 13-14 November 2006, pp 567-573

Jorstad, I., Thanh, D. V., Dustdar, S. (2006). Service Personalisation in Mobile Heterogenous Environments. *In Proceedings of Advanced International Conference on Telecommunications (AICT2006)*, Guadeloupe, French Caribbean , February 19-22, 2006.

Jung, S. Y., Hong, J. H., Kim, T. S. (2003). A Formal Model for Preference, *In proceedings of the 2002 IEEE international Conference on Data Mining (ICDM'02)*, Maebashi TERRSA, Maebashi City, Japan, 9-12 December 2002, pp 235.

Jung, S. Y., Hong, J. H., Kim, T. S. (2005). A Statistical Model for User Preferences, *IEEE Transaction on Knowledge and Data Engineering*, Vol. 17, No 6, June 2005; pp 834-843.

Kantardzic, M. (2001). Data Mining Concepts, Models, Methods, and Algorithms;

IEEE Press.

Kay, J., Kummerfeld, R. J. (2003). Managing Private User Models and Shared Personas; *In: k. Cheverst, B. N. de Carolis & A. Kruger, eds, 'UM03 Workshop on User Modelling for Ubiquitous Computing', 22-23 June 2003.* <http://www.di.uniba.it/~ubium03/kay-4.pdf> (last accessed on 24 April 2006).

Kiessling, W. (2002). Foundations of Preferences in Databases; *In: Proceedings 28<sup>th</sup> International Conference on Very Large Databases (VLDB 2002)*, Hong Kong, China, 20-23 Aug 2002 <http://www.vldb.org/conf/2002/S09P04.pdf> (Last accessed on 24 April 2006)

Kiewera, M. (2005). Iterative Discovering of User's Preferences Using Web Mining; *International Journal of Computer Science & Applications, Vol.II, No.II*, pp 57-66.

Kim, C., Kim, J. (2003). A Recommendation Algorithm Using Multilevel Association Rules, *In Proceedings of the IEEE/WIC International Conference on Web Intelligence (WI'03)*, Beijing, China, 13-17 October 2003, pp 524 – 527

Klan, F. (2006). Context-aware Service discovery, selection and usage; *In Proceedings 18<sup>th</sup> GI-Workshop on the Foundations of Databases*, Wittenberg, Saxony-Anhalt, June 2006, <http://hnsp.inf-bb.uni-jena.de/DIANE/docs/GvD06.pdf>. (Last accessed 29 march 2007)

Klevecz, B. (1999). The Whole EST Catalog" *Scientist* 12 (2): 22 Jan 18 1999

Kobsa, A. (2001). Generic User Modeling Systems. *User Modeling and User-Adapted Interaction* 11(1-2), 49-63. <http://www.ics.uci.edu/~kobsa/papers/2001-UMUAI-kobsa.pdf>. (last 29 March 2007)

Kobsa, A., Fink, J (2003). Performance Evaluation of User Modeling Servers

Under Real-World Workload Conditions. In: P. Brusilovsky, A. T. Corbett and F. de Rosis, eds: *User Modeling 2003: 9th International Conference, UM 2003, Johnstown, PA, Springer Verlag (LNCS)*, pp 143-153. <http://www.ics.uci.edu/~kobsa/papers/2003-UM-kobsa.pdf>. (Last accessed 29 March 2007)

Mamei, M., Zambonelli, F., Leonardi, L. (2003). Tuples On The Air: A Middleware for Context-Aware Computing in Dynamic Networks; In the *1<sup>st</sup> International ICDCS Workshop on Mobile Computing Middleware (MCM03)*, Providence, Rhode Island. May 2003. [http://citeseer.ist.psu.edu/cache/papers/cs/32918/http:zSzzSzzeus.elet.polimi.itzSzis-manetzSzDocument\\_izSzpap-dismi-11.pdf/mamei03tuples.pdf](http://citeseer.ist.psu.edu/cache/papers/cs/32918/http:zSzzSzzeus.elet.polimi.itzSzis-manetzSzDocument_izSzpap-dismi-11.pdf/mamei03tuples.pdf) (Last accessed on 29 June 30, 2006).

Mäntyjärvi, J., Himberg, J., Korpipää, P. & Mannila, H. 2001. Extracting the context of a mobile device user. In *Proceedings of the International Symposium on Human-Machine Systems (HMS)*, Kassel, Germany, pp 445–450.

Montgomery, L. A., Faloutsos, C. (2000). Trends and Patterns of WWW Browsing Behaviour. <http://www.andrew.cmu.edu/user/alm3/papers/web%20trends.pdf> (Last accessed on 29 March 2007)

Niederee, C., Stewart, A., Mehta, B., Hemmje, M. (2004). A Multi-Dimensional, Unified User Model for Cross-System Personalization; In *Proceedings of the Workshop on Environments for Personalized Information Access Working Conference on Advanced Visual Interfaces AVI 2004*; Gallipoli, Italy, May 25, 2004, pp 34-54. [http://www.di.uniba.it/avi2004/e4pia/EPIA2004\\_proceedings.pdf](http://www.di.uniba.it/avi2004/e4pia/EPIA2004_proceedings.pdf) (Last accessed on 23 February 2007)

Pashtan, A. (2004). *Mobile Web Services* (1<sup>st</sup> Edition); Cambridge University Press, The Edinburgh building, Cambridge CB2 2RU, UK

Paulson, P., Tzanavari, A. (2003). Combining Collaborative and Content-Based Filtering Using Conceptual Graphs. *In J.Lawry, J.G.Shanahan and A.Ralescu (eds.): Modeling with Words: Learning, Fusion, and Reasoning within a Formal Linguistic Representation Framework, LNAI 2873, Berlin Heidelberg, pp 168-185*

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J. (1994). Grouplens: An Open Architecture for Collaborative Filtering of Netnews; *In Proceedings of ACM Computer Supported Cooperative Work, Chapel Hill, pp 175-186*

Riva, O. (2004). A Conceptual Model for Structuring Context-Aware Applications; *Forth Berkeley-Helsinki student workshop on telecommunication Software architectures; University of Berkeley USA; 2004.*

Riva, O., Flora, C. (2006). Controy: A Smart Phone Middleware Supporting Multiple Context Provisioning Strategies; *2nd International Workshop on Services and Infrastructure for the Ubiquitous and Mobile Internet (SIUMI'06), at the 26th International Conference on Distributed Computing Systems (ICDCS'06), Lisbon (Portugal), 4-7 July 2006. [www.cs.helsinki.fi/u/riva/publications/riva\\_siumi06\\_paper.pdf](http://www.cs.helsinki.fi/u/riva/publications/riva_siumi06_paper.pdf) (Last accessed on 29 June 2006).*

Schafer, J. B., Konstan, J. A., Riedl, J. (1999). Recommender Systems in E-Commerce; *In Proceedings of the ACM Conference on Electronic Commerce, Denver, Colorado, USA, 3-5 November 1999.*

Schmidt, A., Adoo K. A., Takaluoma, A., Tuomela, U., Laerhoven, K. V., Velde, W. V. (1999). Advanced Interaction in Context; *Lecture Notes in Computer Science, <http://citeseer.ist.psu.edu/cache/papers/cs/12585/http:zSzzSzwww.teco.uni-karlsruhe.de/z/~albrechtzSzpublicationzSzshuc99zSzadvanced%20interaction%20context.pdf/schmidt99advanced.pdf> (last accessed 24 April 2006)*

Shyu, M. L., Haruechaiyasak, C., Chen, S. H., Zhao, N. (2005); Collaborative Filtering by Mining Association Rules from User Access Sequences; *In Proceedings of 2005 International Workshop on Challenges in Web Information Retrieval and Integration (WIRI 2005)*, Tokyo, Japan, 8-9 April 2005, pp 128-135, <http://www.cs.fiu.edu/~chens/PDF/WIRI05.pdf> (Last accessed 29 March 2007)

Sicilia, M. A., Garcia, E. (2004). On the Use of Bipolar Scales in Preference-Based Recommender Systems, *In Proceedings 5th International Conference on Electronic Commerce and Web Technologies (EC-Web 2004)*, Zaragoza, Spain, August 31-September 3, 2004; pp 268-276 [http://citeseer.ist.psu.edu/cache/papers/cs2/93/http:zSzzSzwww.cc.uah.esSzmsiciliazSzpaperszSzSicilia\\_ECWEB\\_2004.pdf/sicilia04use.pdf](http://citeseer.ist.psu.edu/cache/papers/cs2/93/http:zSzzSzwww.cc.uah.esSzmsiciliazSzpaperszSzSicilia_ECWEB_2004.pdf/sicilia04use.pdf) (Last accessed on 12 June 2007)

Tintarev, N., Masthoff, J. (2006). Similarity for News Recommender Systems; <http://www.csd.abdn.ac.uk/~jmasthof/Publications/WPRSIUI06.pdf> (Last accessed on 12 June 2007)

Toivonen, S. (2004). Hybrid service provision model for mobile users: Prospects for the DYNAMOS project. *In Proceedings of the 11<sup>th</sup> Finnish Artificial Intelligence Conference (STeP 2004)*, Vantaa, Finland, September 1-3, 2004, Volume 2. (2004), pp 183-192 <http://virtual.vtt.fi/virtual/proj2/dynamos/pubs/toivonendynamos.pdf> (Last accessed 29 March 2007)

Tompson, M. S., Midkiff, S. F. (2005), Service Description for Pervasive Service Discovery; *In Proceedings of the 25<sup>th</sup> IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'05)*, Columbus, OH, USA, 6-10 June 2005, pp 273 - 279

Tseng, V. S. M., Lin, K. W. C (2005). Mining sequential mobile access patterns efficiently in mobile Web systems; *In Proceedings of the 19<sup>th</sup> International*

*Conference on Advanced Information Networking and Applications, 2005 (AINA 2005)*, Volume 2, 28-30 March 2005, Taipei, TAIWAN; pp762 - 767

Wang, Q., Balke, W., Kiessling, W., Huhn, A. (2004); P-News: Deeply Personalized News Dissemination for MPEG-7 Based Digital Libraries; *In Proceedings of the European Conference on Digital Libraries (ECDL)*, Bath, London, UK, September 12-17, 2004, pp 256-268

Yang, Q., Knoblock, C. A., Wu, X. (2004). Guest Editors' Introduction: Mining Actionable Knowledge on the Web, *Intelligent Systems* Volume 19, Issue 6, Nov.-Dec. 2004, pp 30 – 31

Yang, Y., Williams, H. M., Pooley, R., Dewar, R. (2006). Context-Aware Personalization in Pervasive Communications; *In Proceedings of the IEEE International Conference on e-Business Engineering (ICEBE'06)*, Shanghai, China, October 24-26, 2006, pp. 663-669

Zheng, T., Iyengar, V. S. Recommender Systems Using Linear Classifiers, *Journal of Machine Learning* 2(2002), pp 313-334.

# APPENDIX A

## User Preference Profile DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Title           UserPreferenceProfile
    Description      Document Type Definition for the persistent storage of user preference profiles
    Date            05.01.07
    Author          Edgar Jembere
-->

<!--
    UserPreferenceProfile is the root element
    Profiles can be stored for several users
-->

<!ELEMENT UserPreferenceProfile          (UserIdentifier*)>

<!--
    Each user is identified by the unique name and may have several User Preferences Profiles
-->

<!ELEMENT UserIdentifier                  (UserPreferenceProfileData+)>
<!ATTLIST UserIdentifier                  Value CDATA #REQUIRED>

<!--
    Each user has preferences preference under different domains
-->

<!ELEMENT UserPreferenceProfileData      (Context+)>
<!ATTLIST UserPreferenceProfileData      Value CDATA #REQUIRED>

<!--
    A context is specified by a specific unique key determined by the context model
-->

<!ELEMENT Context                        (TimeStamp, Attribute+)>
<!ATTLIST Context                        Value CDATA #REQUIRED>

<!--
    TimeStamp contains the date and Time when a preference was mined.
-->

<!ELEMENT TimeStamp                      EMPTY>
<!ATTLIST TimeStamp                      dateTime CDATA #REQUIRED>

<!--
    An Attribute has one or more features/ attribute values
-->

<!ELEMENT Attribute                      (AttributeValue+)>
<!ATTLIST Attribute                      Value CDATA #REQUIRED>

<!--
    An Attribute Value is defined by the Attribute Value name (Key) and a preference value (Value)
-->

<!ELEMENT AttributeValue                  EMPTY>
<!ATTLIST AttributeValue                  Key CDATA #REQUIRED>
<!ATTLIST AttributeValue                  Value CDATA #REQUIRED>
```

# APPENDIX B

## Preference Repository DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Title           Preference Repository
    Description      Document Type Definition for the persistent storage of user preferences
    Date            02.11.06
    Author          Edgar Jembere
-->

<!--
    PreferenceRepository is the root element
    Preferences can be stored for several users
-->

<!ELEMENT PreferenceRepository                (UserIdentifier*)>

<!--
    Each user is identified by the unique name and may have several preferences
-->

<!ELEMENT UserIdentifier                    (PreferenceData+)>
<!ATTLIST UserIdentifier                    Value CDATA #REQUIRED>

<!--
    Each PreferenceData (Domain) has one or more context/situation(s) under which a given user's preference can
    be defined
-->

<!ELEMENT PreferenceData                    (Context+)>
<!ATTLIST PreferenceData                    Value CDATA #REQUIRED>

<!--
    A context is specified by a specific unique key determined by the context model. This part of the DTD can be
    adapted to application-specific needs.
-->

<!ELEMENT Context                          (TimeStamp, Preference+)>
<!ATTLIST Context                          Value CDATA #REQUIRED>

<!--
    TimeStamp contains the date and Time when a preference was mined.
-->

<!ELEMENT TimeStamp                        EMPTY>
<!ATTLIST TimeStamp                        dateTime CDATA #REQUIRED>

<!--
    Preference defines a user preference by using the Preference type.
-->

<!ELEMENT Preference (
                                POS
                                | STRONG_NEG
                                | SOFT_NEG
                                | POSPOS
                                | POSSTRONG_NEG
                                | POSSOFT_NEG
```

```

| SOFT_NEGSOFT_NEG
| POSSOFT_NEGSOFT_NEG
| SOFT_NEGSTRONG_NEG
| EXPLICITITEM
| EXPLICITPOS
| EXPLICITSOFT_NEG
| POSEXPLICITSOFT_NEG
| EXPLICIT_POSEXPLICIT_SOFT_NEG
)>

<!--
    POS Preference
    POSSet: Set of positively preferred values/items
-->

<![ELEMENT POS                                (POSSet)>
<![ATTLIST POS                                att CDATA #REQUIRED>
<![ELEMENT POSSet                             (Value+)>

<!--
    Auxiliary definition for the values of a POS-set, SOFT_NEG-set, e.t.c.
-->

<![ELEMENT Value                             EMPTY>
<![ATTLIST Value                             val CDATA #REQUIRED>

<!--
    SOFT_NEG Preference
    SOFT_NEG-set: Set of soft negative values/items
-->

<![ELEMENT SOFT_NEG                          (SOFT_NEGSet)>
<![ATTLIST SOFT_NEG                          att CDATA #REQUIRED>
<![ELEMENT SOFT_NEGSet                       (Value+)>

<!--
    STRONG_NEG Preference
    STRONG_NEG-set: Set of strong negative values/items
-->

<![ELEMENT STRONG_NEG                        (STRONG_NEGSet)>
<![ATTLIST STRONG_NEG                        att CDATA #REQUIRED>
<![ELEMENT STRONG_NEGSet                     (Value+)>

<!--
    POSPOS Preference
    POS1-set: set of positive values
    POS2-set: set of alternative positive values
-->

<![ELEMENT POSPOS                            (POS1Set, POS2Set)>
<![ATTLIST POSPOS                            att CDATA #REQUIRED>
<![ELEMENT POS1Set                           (Value+)>
<![ELEMENT POS2Set                           (Value+)>

<!--
    POSSOFT_NEG Preference
    POS-set: set of positive values
    SOFT_NEG-set: set of soft negative values
-->

<![ELEMENT POSSOFT_NEG                       (POSSet, SOFT_NEGSet)>
<![ATTLIST POSSOFT_NEG                       att CDATA #REQUIRED>

<!--
    POSSTRONG_NEG Preference
    POS-set: set of positive values
    STRONG_NEG-set: set of strong negative values
-->

```

```

<ELEMENT POSSTRONG_NEG                                (POSSet, STRONG_NEGSet)>
<!ATTLIST POSSTRONG_NEG                                att CDATA #REQUIRED>

<!--
    SOFT_NEGSTRONG_NEG Preference
    SOFT_NEG-set: set of soft negative values
    SOFT_NEG-set: set of soft negative values
-->

<ELEMENT SOFT_NEGSOFT_NEG                              (SOFT_NEG1Set, SOFT_NEG2Set)>
<!ATTLIST SOFT_NEGSOFT_NEG                              att CDATA #REQUIRED>
<ELEMENT SOFT_NEG1Set                                  (Value+)>
<ELEMENT SOFT_NEG2Set                                  (Value+)>

<!--
-->
<ELEMENT POSSOFT_NEGSOFT_NEG                          (POSSet, SOFT_NEG1Set, SOFT_NEG2Set)>
<!ATTLIST POSSOFT_NEGSOFT_NEG                          att CDATA #REQUIRED>

<!--
    SOFT_NEGSTRONG_NEG Preference
    SOFT_NEG-set: set of soft negative values
    STRONG_NEG-set: set of alternative soft negative values
-->

<ELEMENT SOFT_NEGSTRONG_NEG                            (SOFT_NEGSet, STRONG_NEGSet)>
<!ATTLIST SOFT_NEGSTRONG_NEG                            att CDATA #REQUIRED>

<!--
    EXPLICITPOSCATEGORICAL: defines preference categories when we have more
    than two(2) POS preference categories
-->

<ELEMENT EXPLICITPOS                                    (CATEGORY+)>
<!ATTLIST EXPLICITPOS                                    att CDATA #REQUIRED>

<ELEMENT CATEGORY                                      (Value+)>
<!ATTLIST CATEGORY                                      att CDATA #REQUIRED>

<!--
    EXPLICITSOFT_NEGCATEGORICAL: defines preference categories when we have more
    than one (1) SOFT_NEG preference categories
-->

<!--
<ELEMENT EXPLICITSOFT_NEG                              (CATEGORY+)>
<!ATTLIST EXPLICITSOFT_NEG                              att CDATA #REQUIRED>

<!--
    POSEXPLICITSOFT_NEG: defines preference categories when we have one (1) POS category and more than one
    (1) POS preference categories
-->

<!--
<ELEMENT POSEXPLICITSOFT_NEG                          (POSSet,CATEGORY+)>
<!ATTLIST POSEXPLICITSOFT_NEG                          att CDATA #REQUIRED>

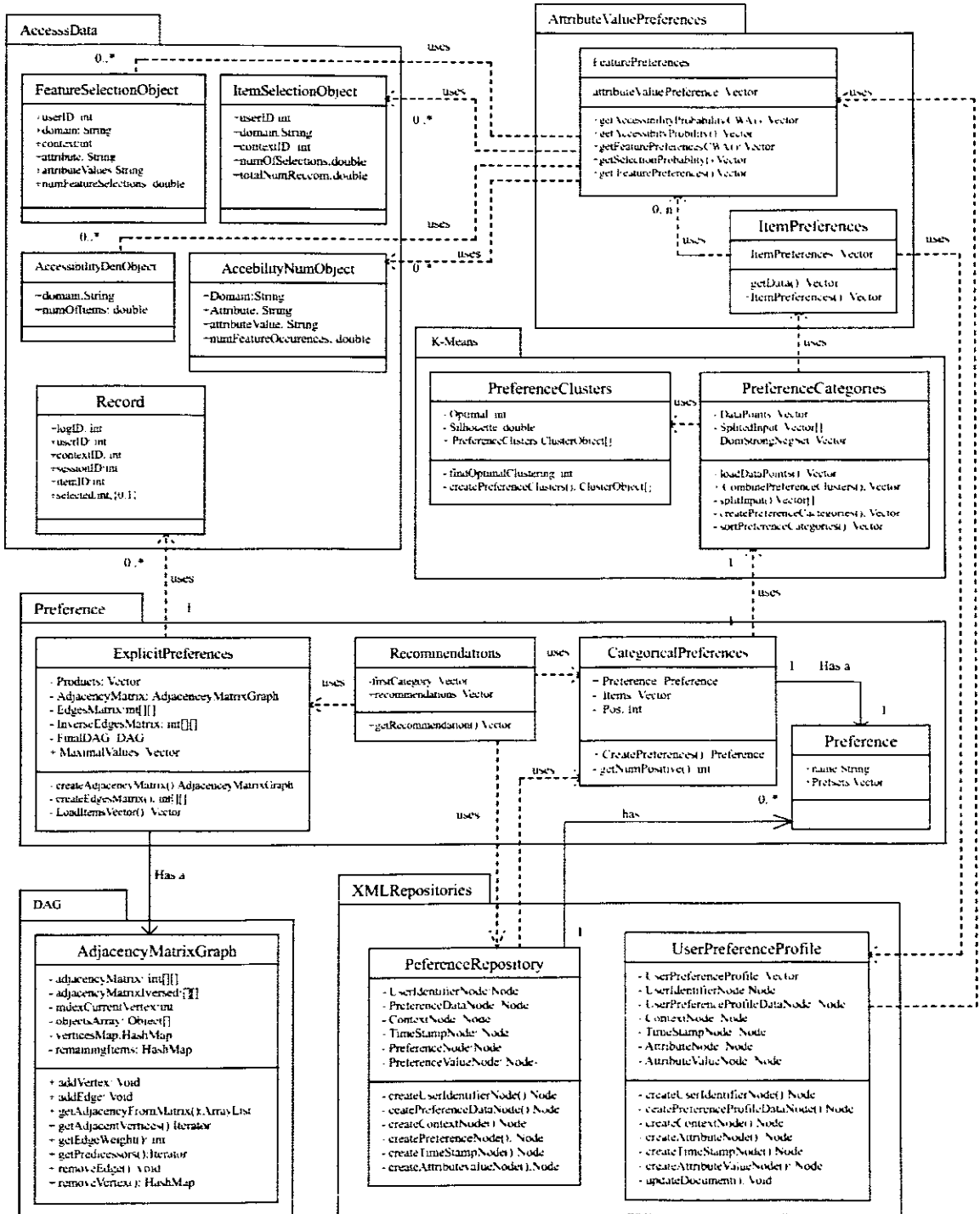
<!--
    EXPLICITPOSEXPLICITSOFT_NEG: defines preference categories when we have more thanone (1) POS category
    and more than one (1) POS preference categories
-->

<!--
<ELEMENT EXPLICITPOSEXPLICITSOFT_NEG                  (CATEGORY+, CATEGORY+)>
<!ATTLIST EXPLICITPOSEXPLICITSOFT_NEG                  att CDATA #REQUIRED>

```

# APPENDIX C

## Class Diagram



# APPENDIX D

## Formalisation of the Categorical Preferences as Strict Partial orders

This section serves to show that the presented user preferences representation is strict partial order. Two properties of strict partial order preferences presented in (Kiessling, 2002), hierarchical properties and duality of preferences will be presented.

### A.D.1 Preferences Hierarchies

From the preference framework presented in (Kiessling, 2002) preferences can be arranged in hierarchies. Given constructors  $C1$  and  $C2$ ,  $C1$  is a preference sub-  
constructor of  $C2$  ( $C1 \leq C2$ )<sup>1</sup>, if the definition of  $C1$  can be obtained from the definition of  $C2$  by some specializing constraints.

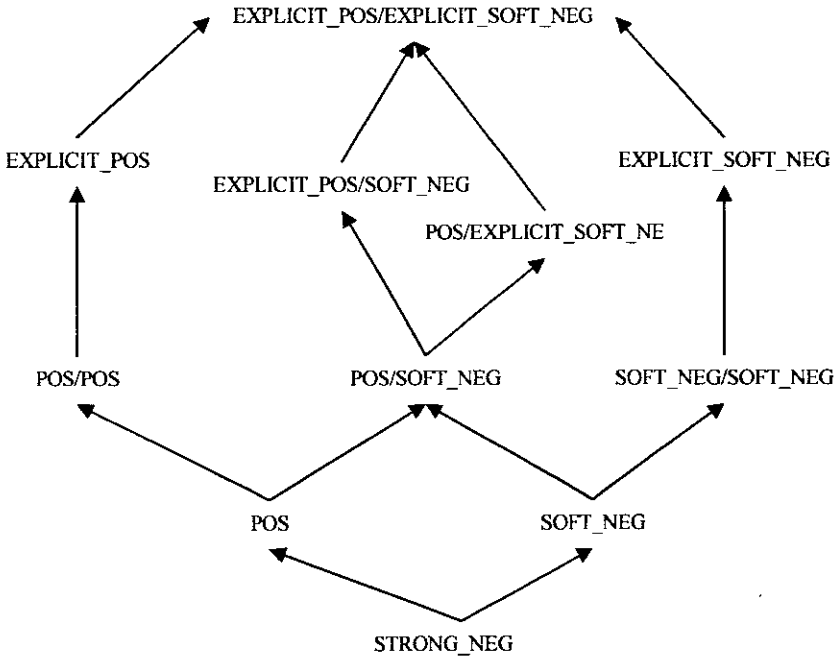


Figure D. 1: Preference Hierarchies

<sup>1</sup>  $\leq$  denotes “is a sub-  
constructor of” and not “less than or equal to”

Based on this argument the preference hierarchy of base preference constructors in Figure D.1 is derived as follows:

- $STRONG\_NEG \leq POS$  , if  $POS - set = \emptyset$  .
- $STRONG\_NEG \leq SOFT\_NEG$  , if  $SOFT\_NEG - set = \emptyset$  .
- $POS \leq POS / POS$  , if  $POS2 - set = \emptyset$  .
- $POS \leq POS / SOFT\_NEG$  , if  $SOFT\_NEG - set = \emptyset$  .
- $SOFT\_NEG \leq SOFT\_NEG / SOFT\_NEG$  , if  $SOFT\_NEG2 - set = \emptyset$  .
- $SOFT\_NEG \leq POS / SOFT\_NEG$  , if  $POS - set = \emptyset$  .
- $POS / POS \leq EXPLICIT\_POS$  , if  $EXPLICIT_p - graph = (POS1 - set)^{\leftrightarrow} \oplus (POS2 - set)^{\leftrightarrow 2}$  .
- $SOFT\_NEG / SOFT\_NEG \leq EXPLICIT\_SOFT\_NEG$  , if  $EXPLICIT_{SN} - graph = (SOFT\_NEG1 - set)^{\leftrightarrow} \oplus (SOFT\_NEG2 - set)^{\leftrightarrow}$  .
- $POS / SOFT\_NEG \leq POS / EXPLICIT\_SOFT\_NEG$  , if  $EXPLICIT_{SN} - graph = (SOFT\_NEG - set)^{\leftrightarrow}$  .
- $POS / SOFT\_NEG \leq EXPLICIT\_POS / SOFT\_NEG$  , if  $EXPLICIT - graph = (POS - set)^{\leftrightarrow}$  .
- $POS / EXPLICIT\_SOFT\_NEG \leq EXPLICIT\_POS / EXPLICIT\_SOFT\_NEG$  , if  $EXPLICIT_p - graph = (POS - set)^{\leftrightarrow}$  .
- $EXPLICITPOS / SOFT\_NEG \leq EXPLICIT\_POS / EXPLICIT\_SOFT\_NEG$  , if  $EXPLICIT_{SN} - graph = (SOFT\_NEG - set)^{\leftrightarrow}$  .
- $EXPLICITPOS \leq EXPLICIT\_POS / EXPLICIT\_SOFT\_NEG$  , if  $EXPLICIT_{SN} - graph = \emptyset$  .
- $EXPLICITSOFT\_NEG \leq EXPLICIT\_POS / EXPLICIT\_SOFT\_NEG$  , if  $EXPLICIT_p - graph = \emptyset$  .

## A.D.2 Duality of Preferences

According to the preference algebra presented in (Kiessling, 2002), Strict Partial order preferences have the duality property.

---

<sup>2</sup>  $\oplus$  denotes the linear sum preferences and  $\leftrightarrow$  marks an anti-chain preferences as defined in (Kiessling, 2002)

**Definition A.D.1**

For two values  $a$  and  $b$  the dual data-driven preference  $a \prec_{PD}^{\partial} b$  holds if and only if  $b \prec_{PD} a$ .

**Proposition A.D.1**

For the data-driven preferences POS and SOFT\_NEG, we have the following duality correlations:

$$POS^{\partial} = SOFT\_NEG \text{ and } SOFT\_NEG^{\partial} = POS$$

**Proof**

$POS^{\partial}$  denotes that all values of POS-Set have negative preference values.  $\therefore$ , we have a SOFT\_NEG preference

**Proposition A.D.2**

For the data-driven preferences POS/POS and SOFT\_NEG/SOFT\_NEG, we have the following duality correlations:

$$POS/POS^{\partial} = SOFT\_NEG/SOFT\_NEG \text{ and } SOFT\_NEG/SOFT\_NEG^{\partial} = POS/POS$$

**Proof**

$POS/POS^{\partial}$  denotes that all values of  $POS1-Set$  have lesser negative preference values than all value of  $POS2-Set$ . This implies that we have  $SOFT\_NEG1-Set(POS2-Set)$  and  $SOFT\_NEG2-Set(POS1-Set)$ . Therefore we have  $SOFT\_NEG/SOFT\_NEG$

**Proposition A.D.3**

A POS/SOFT\_NEG is dual to itself. Thus we have the following duality relation:

$$POS/SOFT\_NEG^{\partial} = POS/SOFT\_NEG$$

**Proof**

$POS / SOFT\_NEG^{\circ}$  denotes that all values of  $POS - Set$  have negative preference values and all the values of  $SOFT\_NEG - set$  have positive values . Therefore we a  $POS / SOFT\_NEG$  preference

**Proposition A.D.4**

For the data-driven preferences  $POS / EXPLICITSOFT\_NEG$  and  $EXPLICTPOS / SOFT\_NEG$ , we have the following duality correlations:

$$POS / EXPLICITSOFT\_NEG^{\circ} = EXPLICTPOS / SOFT\_NEG \qquad \text{and} \\ EXPLICTPOS / SOFT\_NEG^{\circ} = POS / EXPLICTITSOFT\_NEG$$

**Proof**

$POS / EXPLICITSOFT\_NEG^{\circ}$  denotes that all values of  $POS - Set$  have negative preference values, CATEGORY1 have the least positive values followed by CATEGORY2 then CATEGORY3 up to CATEGORYN, which have values with the highest positive preference values. Therefore, we have  $EXPLICTPOS / SOFT\_NEG$

**Proposition A.D.5**

A  $EXPLICTPOS / EXPLICITSOFT\_NEG$  preference is dual to itself. Thus we have the following duality relation:

$$EXPLICTPOS / EXPLICITSOFT\_NEG^{\circ} = EXPLICTPOS / EXPLICITSOFT\_NEG$$

**Proof**

The proof of this proposition follows from the proof of Proposition A.D.4 by replacing the POS set with n EXPLICTPOS sets and the SOFT\\_NEG set with m EXPLICITSOFT\\_NEG sets, where n = 2, 3, 4... and m= 2, 3, 4....