



Development of an Autonomous Distributed MultiAgent Monitoring System for the Automatic Classification of End Users

A thesis submitted to the Faculty of Advanced Technology of the University of

Glamorgan By

Mohammed Ramadan Mhereeg

In partial fulfilment of the requirements for the degree of Doctor of Philosophy

May 2011

Certificate of Research

This is to certify that, except where specific reference is made, the work presented in this thesis is the result of the investigation undertaken by the candidate.

Signed: (Candidate)

Signed: (Director of Studies)

Date:

Declaration

This is to certify that neither this thesis nor any part of it has been presented or is being currently submitted in candidature for any other degree other than the degree of Doctor of Philosophy of the University of Glamorgan.

Signed: (Candidate)

Date:

I hereby give consent for my thesis, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organizations.

Signed: (Candidate)

Date:

I dedicate this thesis
To
My wife and our daughter,
My parents,
And
My parents-in-law

Abstract

The purpose of this study is to investigate the feasibility of constructing a software Multi-Agent based monitoring and classification system and utilizing it to provide an automated and accurate classification for end users developing applications in the spreadsheet domain. Resulting in, is the creation of the Multi-Agent Classification System (MACS).

The Microsoft's .NET Windows Service based agents were utilized to develop the Monitoring Agents of MACS. These agents function autonomously to provide continuous and periodic monitoring of spreadsheet workbooks by content. .NET Windows Communication Foundation (WCF) Services technology was used together with the Service Oriented Architecture (SOA) approach for the distribution of the agents over the World Wide Web in order to satisfy the monitoring and classification of the multiple developer aspect. The Prometheus agent oriented design methodology and its accompanying Prometheus Design Tool (PDT) was employed for specifying and designing the agents of MACS, and Visual Studio.NET 2008 for creating the agency using visual C# programming language.

MACS was evaluated against classification criteria from the literature with the support of using real-time data collected from a target group of excel spreadsheet developers over a network. The Monitoring Agents were configured to execute automatically, without any user intervention as windows service processes in the .NET web server application of the system. These distributed agents listen to and read the contents of excel spreadsheets development activities in terms of file and author properties, function and formulas used, and Visual Basic for Application (VBA) macro code constructs. Data gathered by the Monitoring Agents from various resources over a period of time was collected and filtered by a Database Updater Agent residing in the .NET client application of the system. This agent then transfers and stores the data in Oracle server database via Oracle stored procedures for further processing that leads to the classification of the end user developers.

Oracle data mining classification algorithms: Naive Bayes, Adaptive Naive Bayes, Decision Trees, and Support Vector Machine were utilized to analyse the results from the data gathering process in order to automate the classification of excel spreadsheet developers. The accuracy of the predictions achieved by the models was compared. The results of the comparison showed that Naive Bayes classifier achieved the best results with accuracy of 0.978. Therefore, the MACS can be utilized to provide a Multi-Agent based automated classification solution to spreadsheet developers with a high degree of accuracy.

Acknowledgment

I would like to express my sincere appreciation and deepest thanks to my supervisors Duncan McPhee, Stephen Hole, and Paul Angel for their continual assistance, patience, and guidance throughout the duration of this thesis.

I would also like to thank the Faculty's IS Customer Support (LCSS) department that permitted, and facilitated arrange the data collection and evaluation activities.

A very special thank you to my wife and family whose love, support, and devotion helped to make this work possible.

Last but not least, I am thankful to my friends and colleagues for their fruitful encouragement and motivation.

Published Works

This work documented in this thesis has resulted in a number of published research papers.

- MHEREEG, M. & MCPHEE, D. & HOLE, S. & NATHAN, T. (2010) Design and Implementation of Autonomous Classification System using .NET Windows Service Agent Technology. *6th International Conference on Computer Science and Information Systems*. Athens, Greece, Athens Institute for Education and Research.
- MHEREEG, M. & MCPHEE, D. & HOLE, S. & ANGEL, P. (2009), Automatic Classification and Graphical Representation of Interactive User Activity via .NET Windows Service Agent Technology, *Fifth International Conference on Autonomic and Autonomous Systems, IEEE, Spain, , ICAS*, pp.94-99.
- MCPHEE, D. & MHEREEG, M. & HOLE, S. & NATHAN, T. (2009), Using a Software Multi-Agent System to Assess Student Learning Outcomes. *4th Plymouth e-learning Conference*. Plymouth, UK.
- HOLE, S. & MCPHEE, D. & MHEREEG, M. (2008) Developing a Windows Service Agent to Analyze Spreadsheet Macro and Function Complexity. *4th International Conference on Computer Science and Information Systems*. Athens, Greece, Athens Institute for Education and Research.

Table of Contents

- Abstract.....	v
- Acknowledgement.....	vi
- Published Works.....	vii
- List of Figures.....	xiii
- List of Tables.....	xv
- List of Code Segments.....	xvi
- Glossary of Acronyms.....	xvii
 Chapter 1 – Introduction.....	 1
1.1 Aims and objectives of the project.....	1
1.1.1 Aims.....	1
1.1.2 Objectives.....	1
1.2 Research problem.....	2
1.3 The proposed solution to the problem.....	3
1.4 Contribution to knowledge.....	3
1.5 Organization of the thesis.....	3
 Chapter 2 - Background and related work.....	 5
2.1 Introduction.....	5
2.2 Background.....	5
2.2.1 Agent Technology.....	5
2.2.1.1 What is an agent?	5
2.2.1.2 Software Agents.....	6
2.2.1.3 What makes software agents distinctive?	8
2.2.1.4 Agent Usage.....	10
2.2.1.5 How Agents Communicate.....	10
2.2.1.5.1 Agent Communication Languages.....	11
2.2.1.5.2 Agent Transportation Mechanisms.....	12
2.2.1.6 Software Development Methodologies for Agent-Based Systems...	12
2.2.1.6.1 Agile Methodologies	13
2.2.1.6.2 The Rational Unified Process (RUP)	14

2.2.1.6.3 Shlaer-Mellor Method	14
2.2.1.6.4 The Prometheus Methodology.....	15
2.2.1.7 Multi-Agent Systems.....	17
2.2.1.8 Compatibility in Multi-Agent Systems.....	18
2.2.1.8.1 Different Approaches of Standardizations.....	19
2.2.1.8.2 Overview of FIPA Architecture.....	21
2.2.2 .NET Framework.....	22
2.2.2.1 Agent Based Windows Service.....	23
2.2.2.2. .NET Windows Communication Foundation.....	24
2.2.2.3 Hosting WCF Services in a Windows Service Application.....	26
2.2.2.4 Justification of the .NET WCF Technology as the Distribution Mechanism	27
2.2.3 Data Mining.....	30
2.2.3.1 Mining with Oracle Data Mining.....	31
2.2.3.2 Classification Algorithms.....	31
2.2.3.2.1 Naïve Bayes.....	32
2.2.3.2.2 Decision Trees.....	33
2.2.3.2.3 Adaptive Bayes Network.....	34
2.2.3.2.4 Support Vector Machine.....	35
2.2.3.3 Clustering Algorithms	36
2.3 Related Work.....	39
2.3.1 End User Computing.....	39
2.3.2 End User Classification.....	39
2.3.2.1 Different Approaches of Classification.....	40
2.3.2.2 End Users Programmers Classification	42
2.3.2.3 Spreadsheet Technology.....	44
2.3.2.3.1 Spreadsheet Complexity.....	45
2.3.2.3.2 Excel Developer Categories.....	46
2.3.2.3.3 Justification of Spreadsheets as a Workbench for Classification	49
2.4 Research Ethics	50
Chapter 3 - The MultiAgent System Architecture of MACS.....	54
3.1 Introduction.....	54

3.2 MultiAgent System Based on SOA.....	54
3.3 MACS Multi Agent System.....	56
3.3.1 .NET Widows Services based agents as WCF Service Hosts.....	58
3.3.2 Agent Communications in MACS.....	59
Chapter 4 - Design of the .NET Software Agents of MACS.....	60
4.1 System Specification.....	60
4.1.1 Goal Specification.....	60
4.1.1.1 Brief Statement of the Problem Scenario.....	60
4.1.1.2 Identifying Initial Goals.....	61
4.1.1.2.1 Identify System Goals.....	61
4.1.1.2.2 Goal Refinement.....	61
4.1.2 Roles.....	63
4.1.3 Interface Description.....	68
4.1.3.1 Percepts and actions.....	68
4.1.3.2 Data.....	68
4.2 Architectural Design.....	70
4.2.1 Deciding on the Agent Types.....	70
4.2.1.1 Grouping Roles.....	70
4.2.1.1.1 Data Coupling Diagram.....	71
4.2.1.1.2 Agent-Role Grouping Diagram.....	72
4.2.1.1.3 Agent Acquaintance Diagram.....	73
4.2.2 Developing Agent Descriptors.....	73
4.2.3 System Overview.....	75
4.3 Detailed Design.....	76
4.3.1 Agent Overview Diagram.....	76
Chapter 5 – Implementation and Testing	79
5.1 Introduction.....	79
5.2 Implementation of MACS prototype.....	79
5.2.1 The Server-side.....	80
5.2.1.1 The Monitoring Agent.....	81
5.2.1.2 Detecting File Changes.....	89
5.2.2 The Client-Side.....	90

5.2.2.1 The DUA agent.....	92
5.2.2.2 The Service Facilitator (SF).....	94
5.2.2.3 Oracle Server Database 10g Release 10.2.....	95
5.2.3 Implementation Considerations.....	96
5.3 Testing of MACS	97
5.3.1 Testing the Server-Side	97
5.3.2 Testing the Client-Side	98
Chapter 6 – Evaluation and Results.....	103
6.1 Data Gathering.....	103
6.1.1 Data Gathering Strategy	107
6.2 Classification.....	113
6.2.1 Oracle Data Mining Prediction.....	113
6.2.1.1 Algorithms for Classification.....	113
6.2.1.1.1 Problem Definition.....	114
6.2.1.1.2 Data Acquisition and Preparation.....	118
6.2.1.1.2.1 Attribute Importance	118
6.2.1.1.2.1.1 Data Preparation and Ranking	119
6.2.1.1.3 Building and Testing of Models.....	123
6.2.1.1.3.1 Naïve Bayes Predictions.....	123
6.2.1.1.3.2 Adaptive Bayes Network Predictions.....	130
6.2.1.1.3.3 Decision Tree Predictions.....	135
6.2.1.1.3.4 Support Vector Machine Predictions.....	139
6.2.1.1.3.5 Evaluation of the Classification Models’ Predictions.....	143
6.2.1.1.4 The Apply Activity.....	145
Chapter 7 - Conclusion and Future Work.....	150
References.....	158
Appendices.....	170
Appendix A: Agent Properties.....	170
Appendix B: The MA’s Code.....	173
Appendix B1: FileProperties.cs Code.....	173
Appendix B2: Functions.cs Code.....	176

Appendix B3: CodeConstructs.cs Code.....	185
Appendix B4: RegularExpressions.cs Code.....	188
Appendix B5: app.config Configuration File.....	190
Appendix C: Excel Spreadsheet Components/Features.....	192
Appendix D: The DUA's Code.....	194
Appendix D1: The generatedProxyStreamTCP.cs Client Proxy.....	194
Appendix D2: App.config Configuration File.....	196
Appendix E: The Outcome of the Binning Process for Naïve Bayes Model.....	197
Appendix F: The Naïve Bayes Apply Activity Results.....	198

List of Figures

Figure 2.1 Traditional Client-Server approach.....	9
Figure 2.2 software agent based approach.....	9
Figure 2.3 The phases of Prometheus methodology.....	16
Figure 2.4 Agent Management Reference Model.....	21
Figure 2.5 Same/cross-machine communication using WCF.....	25
Figure 2.6 WCF primary components.....	25
Figure 2.7 End-Point Architecture.....	26
Figure 2.8 Typical decision trees for the concept <i>buys_computer</i>	33
Figure 2.9 Linearly separable training data.....	35
Figure 2.10 Grouping of Data Items into Three Clusters	37
Figure 2.11 The User Cube.....	41
Figure 3.1 Service Oriented Architecture Schema.....	54
Figure 3.2 Architecture Skeleton of the MACS Multi-Agent Based System	55
Figure 3.3 MACS Architecture.....	57
Figure 4.1 Goal Overview Diagram.....	63
Figure 4.2 System Roles Diagram.....	64
Figure 4.3 Data Coupling Diagram.....	72
Figure 4.4 Agent-Role Grouping Diagram.....	72
Figure 4.5 Agent Acquaintance Diagram.....	73
Figure 4.6 the System Overview Diagram.....	76
Figure 4.7 Agent Overview Diagram for the Monitoring Agent – Detailed Design.....	77
Figure 4.8 Agent Overview Diagram for the Database Updater Agent – Detailed Design.....	78
Figure 5.1 The Basic Algorithm of MA.	82
Figure 5.2 the Svcutil.exe command.....	91
Figure 5.3 the GUI of MACS.....	91
Figure 5.4 the basic algorithm of the DUA.....	94
Figure 6.1 Grouping of Functions used by Microsoft Excel 2007	104
Figure 6.2 Excel Macro Security Environment Settings.....	106

Figure 6.3 Sample Build Data for Attribute Importance	120
Figure 6.4 The Histogram of the Predictor Attributes	121
Figure 6.5 The Ranks of the Predictor Attributes	122
Figure 6.6 Enabling the Discretizing Step	124
Figure 6.7 The Binning of the Data Performed by Naïve Bayes Model Building..	125
Figure 6.8 predictive confidence of Naïve Bayes algorithm.....	126
Figure 6.9 Accuracy of Naïve Bayes Algorithm.....	127
Figure 6.10 Cost of the predictions of Naïve Bayes model.....	127
Figure 6.11 the Confusion Matrix of Naïve Bayes model.....	128
Figure 6.12 the totals and costs of Naïve Bayes Confusion Matrix.....	129
Figure 6.13 predictive confidence of Adaptive Bayes Network algorithm.....	130
Figure 6.14 Accuracy of Adaptive Bayes Network Algorithm.....	131
Figure 6.15 Cost of the predictions of Adaptive Bayes Network model.....	131
Figure 6.16 the Confusion Matrix of Adaptive Bayes Network model.....	132
Figure 6.17 the totals and costs of Adaptive Bayes Network Confusion Matrix....	133
Figure 6.18 Rules generated for Single-Feature Build.....	134
Figure 6.19 predictive confidence of Decision Tree algorithm.....	135
Figure 6.20 Accuracy of Decision Trees Algorithm.....	136
Figure 6.21 Cost of the predictions of Decision Tree model.....	136
Figure 6.22 the Confusion Matrix of Decision Tree model.....	137
Figure 6.23 the totals and costs of Decision Tree Confusion Matrix.....	138
Figure 6.24 predictive confidence of Support Vector Machine algorithm.....	139
Figure 6.25 Accuracy of Support Vector Machine Algorithm.....	140
Figure 6.26 Cost of the predictions of Support Vector Machine model.....	140
Figure 6.27 the Confusion Matrix of Support Vector Machine model.....	141
Figure 6.28 the totals and costs of Support Vector Machine Confusion Matrix.....	142

List of Tables

Table 2.1 Most prominent Agent Communication Languages.....	11
Table 2.2 General Features of Agile Methods	13
Table 2.3 High-level performance comparison between WCF and existing Microsoft .NET distributed technologies.....	27
Table 2.4 Types of End Users.....	41
Table 2.5 Software Application Usage by US Workers in 2001.....	43
Table 2.6 Excel developer categories.....	48
Table 2.7 Checklist of Requirements for Informed Consent.....	51
Table 4.1 Software usage and object & code patterns.....	69
Table 4.2 the Monitoring Agent Descriptor.....	74
Table 4.3 The Database Updater Agent Descriptor.....	75
Table 5.1 Sample of 20 Records of the Collected Testing Data.....	99
Table 6.1 Summary of the Functions' Grouping.....	105
Table 6.2 Sample Data for Participants' Spreadsheet Development.....	109
Table 6.3 Excel Developer Types Ranking.....	114
Table 6.4 Excel Developer categories.....	116
Table 6.5 Naïve Bayes Confusion Matrix analysis.....	128
Table 6.6 Adaptive Bayes Network Confusion Matrix analysis.....	132
Table 6.7 Decision Tree Confusion Matrix analysis.....	137
Table 6.8 Support Vector Machine Confusion Matrix analysis.....	141
Table 6.9 Evaluation of the classification models' predictions.....	145
Table 6.10 Expectations of the Expert for the Cases of the Apply Data Set.....	146
Table 6.11 The Naïve Bayes apply activity results.....	148
Table 6.12 the analysis of Naïve Bayes apply activity results.....	148

List of Code Segments

Code 5.1 WCF Service Contract and the Service Class that implements it.....	83
Code 5.2 Checking a WCF service host.....	84
Code 5.3 Opening a spreadsheet.....	85
Code 5.4 Stop MA Agents.....	86
Code 5.5 app.config Configuration File.....	88
Code 5.6 FileSystemWatcher Initialization.....	90
Code 5.7 Asynchronous Callback to the service J345_mm.....	92
Code 5.8 Querying the SF for registered services.....	95

Glossary of Acronyms

ABN - Adaptive Bayes Network
ACL - Agent Communication Language
ADO - ActiveX Data Objects
AMS - Agent Management System
API - Application Programming Interface
AP - Agent Platforms
ARCOL - ARTIMIS Communication Language
ASMX - Active Server Methods Extension
ASP - Active Server Pages
CART - Classification and Regression Trees
CBIS - Computer Based Information System
CLR - common language runtime
CoLa - Communication and Coordination Language
COM - Component Object Model
COOL - Domain independent COOrdination Language
CORBA - Common Object Request Broker Architecture
DAI - Distributed Artificial Intelligence
DCD - Document Content Definition
DCOM - Distributed Component Object Model
DF - Directory Facilitator
DLL - Dynamic Link Libraries
DPS - Distributed Problem Solving
DUA - Database Updater Agent
EM - Expectation-Maximization
EUC - End Users Computing
FIPA - Foundation for Intelligent Physical Agents
ACL - Agent Communication Language
GUI - Graphical User Interface
HTTP - Hyper Text Transfer Protocol
ICL - Internet Communication Language
IEUC - Institute of End User Computing
IIS - Internet Information Services
ISC - Internet Server Connection

KIF - Knowledge Interchange Format
KNN - K-Nearest Neighbours
KQML - Knowledge Query Meta Language
MA - Monitoring Agents
MACS - Multi-Agent Classification System
MAS - Multi Agent Systems
MaSE - Multiagent System Engineering
MASIF - Mobile Agent System Interoperability Facility
MMC - Microsoft Management Console
MSMQ - Microsoft Message Queuing
MTS - Message Transport Service
NB - Naive Bayes
ODBC - Object Database Connectivity
ODM - Oracle Data Mining
ODMr - Oracle Data Miner
ODP - Oracle Data Provider
OMG - Object Management Group
PDT - Prometheus Design Tool
PWC - Price, Waterhouse and Coopers
RDF - Resource Description Framework
RFI - Request For Information
RMI - Remote Method Invocation
RPC - Remote Procedure Call
SDML - Strictly Declarative Modelling Language
SOA - Service Oriented Architecture
SOAP - Simple Object Access Protocol
STL - Simple Thread Language
SVM - Support Vector Machine
TCP - Transmission Control Protocol
UA - User Agent
URL - Uniform Resource Locator
VBA - Visual Basic for Application
WAS - Windows Activation Service
WCF - Windows Communication Foundation
WPF - Windows Presentation Foundation

WSE - Web Services Enhancements
WSE - Web Services Enhancements
XLM - Extended Learning Model
XML - eXtensible Markup Language
XP - Extreme Programming
FDD - Scrum, Feature Driven Development
DSDM - Dynamic System Development Method
ASD - Adaptive Software Development
RUP - Rational Unified Process
UML - Unified Modelling Language
OOA - Object-Oriented Analysis
RD - Recursive Design
AI - Attribute Importance
MDL - Minimum Description Length

Chapter 1 - Introduction

This chapter illustrates the aim and objectives of the project, the research problem and the proposed solution to the problem, and the contributions of this work to knowledge. The chapters are then outlined, describing the structure of the thesis.

1.1 Aim and Objectives of the Research Project

1.1.1 Aim

The project aims to construct a distributed Multi-Agent based monitoring and classification system and utilise it to provide automatic and accurate classification for end users developing applications in the spreadsheet domain.

1.1.2 Objectives

1. To consider the role that software multi-agents play within the context of a user classification multi-agent.
2. To identify the agents needed to comprise the multi-agent system required to develop a classification agency.
3. To design the required agents using an appropriate methodology.
4. To develop a prototype multi-agent using an appropriate programming language.
5. To identify an appropriate end user working environment in order to evaluate the classification system.
6. To utilize the system to automatically monitor and categorise end users development activities.
7. To autonomously run the multi-agent system locally as a .NET Windows Services process.
8. To distribute the agents over the World Wide Web using .NET Windows Communication Foundation (WCF) services technology, that is to satisfy the multiple end-users monitoring and classification approach.
9. To identify an efficient communication method for data exchange between agents.

10. To utilize Oracle Data Mining classification algorithms to achieve the classification phase of the system.
11. To determine criteria for the effective evaluation of the Multi-Agent classification system.

1.2 Research Problem

Various methods for classification of End Users Computing (EUC) activities have been proposed in the literature [23] [25] [44] [90] [130]. Although these methods provide classification solutions of end users, these classification schemes rely on the use of questionnaires both paper-based and web-based or interview techniques to gather the necessary data for analysis. These gathering techniques suffer from a number of problems such as validity (the degree to which the gathered data is correct or true, and represent the development activities of a user precisely) and reliability (the degree to which the gathered data can be relied on to categorize end users). Additionally, classification techniques normally require revisiting applications users have been developing in order to monitor and record any changes to knowledge and skills users gain and improve over a period of time, this is a time and effort consuming when it is managed manually, especially if the process requires monitoring and data collection for a long period of time. Classification results could suffer from lack of accuracy (the capability of providing the correct measurement or classification) when large amounts of data are involved.

This project proposes a new automated and distributed data gathering and classification system based on the software Multi-Agent technology. This technique will save time and effort required to complete and return questionnaires and avoids going through unnecessary interviews. The .NET Windows Service based agents will be utilized to automate the data gathering process that leads to the classification of end user developers. The .NET WCF services will be used for the distribution of the agents over a network to satisfy the monitoring and classification of multiple users approach. Spreadsheet applications will be used as end users working environment in order to evaluate the system.

The agent based system is an attempt to remove the organisation's reliance on the human collection of data typically required by most business applications, and replace it with a tool that is capable of providing automatic, consistent and accurate data gathering and

classification of spreadsheet developers. The system could bring benefits of enhanced organizational productivity and cost effectiveness by automating the monitoring, gathering and analysis of data.

The question that results is: is it feasible to automatically and autonomously monitor spreadsheet developers over a network based on the utilization of the software Multi-Agent technology in such a way that makes management capable to accurately classify spreadsheet developers and allow for precise tailor training activities for future spreadsheet development.

1.3 The Proposed Solution to the Problem

The solution to the problem is to develop a distributed Multi-Agent based monitoring tool that automatically and autonomously gather data from multiple spreadsheet developers over a period of time and then create a set of data mining applications that automate the classification process.

1.4 Contribution to Knowledge

The research resulted in three contributions to knowledge in the research areas of End User Computing, Multi-Agent Systems, Distributed Programming, Spreadsheet Development, and Data Mining.

1. The development of a Software Multi-Agent based system that has the capability to automatically and autonomously monitor Microsoft Excel spreadsheets by content.
2. The Utilization of Microsoft's .NET WCF services technology together with SOA architecture for the distribution of the agents over a network in order to satisfy the monitoring of the multiple developers aspect.
3. The utilization of Oracle Data Mining algorithms to automatically and accurately achieve the classification of Excel spreadsheet developers.

1.5 Organization of the Thesis

Chapter 2 is divided into two main subchapters to cover the background and related work of this research. The first subchapter presents the required infrastructure, terminology, concepts and, methodologies of the key areas on which this thesis is focused on with reference to agent

technology and Multi Agent Systems (MAS), efforts on agent standardisation with emphasis to FIPA specifications. The chapter then goes on to review some of the techniques of Microsoft .NET Framework technology. Oracle data mining classification algorithms are also discussed. The second subchapter provides a review of the most important attempts to classify end-user developers, with emphasis to Excel developers' classification.

Chapter 3 describes the agent-based architecture of MACS Classification System. It defines the positions and roles of each agent of the system, and discusses how the agents communicate over a network to achieve their goals.

Chapter 4 presents in detail how Prometheus methodology and its accompanying Prometheus Design Tool (PDT) have been used for specifying and designing the agents of the Multi-Agent Classification System (MACS). This chapter discusses how Prometheus through its three phases, System Specification, Architectural Design and Detailed Design can be employed to achieve this goal.

Chapter 5 presents the implementation and testing of MACS prototype. However, it is divided to four subchapters. The first subchapter presents the software applications used for the development of the prototype. The second subchapter covers the server-side application of MACS. The client-side application is covered in the third subchapter, and some implementation considerations are discussed in the fourth subchapter.

Chapter 6 presents the results obtained from the analysis of the data gathered by the agents of the system. It shows the utilization of Oracle data mining classification algorithms to undertake the task of the analysis in order to achieve the goal of the automatic classification of excel spreadsheet developers.

Chapter 7 concludes the thesis and provides a summary of how the objectives have been accomplished. The contributions of this work to knowledge and future work are also discussed.

Chapter 2 - Background and Related Work

2.1 Introduction

This chapter is divided into two main sections (2.2 and 2.3) to cover the theory and related work of this research respectively. Section 2.2 presents the required infrastructure, terminology, concepts and, methodologies of the key areas on which this thesis is focused on with reference to agent technology and Multi Agent Systems (MAS), and efforts on agent standardisation with emphasis to FIPA specifications. The chapter then goes on to review some of the techniques of .NET Framework technology. Oracle data mining as a means of classification technique is also discussed. Section 2.3 provides a review of the most important attempts to classify end-user developers, with emphasis to Excel developers' classification.

2.2 Background

The theoretical part of this chapter begins with what an agent is, how agents communicate to achieve a goal and, what makes software agents distinctive for this research. Agents are introduced as an emerging technology which can provide a solution not only in the field of end user computing, and in particular the end user classification, but in other areas too, especially due to their autonomous, intelligent, interactive and adaptive characteristics. The importance of a means of distribution and communication within an agent-based system composed of multiple agents with different roles and characteristics, as well as the ability of agents to cooperate with agents from external machine(s) is defined via the utilization of the techniques .NET framework provides. Finally, the need of a classification method that processes the outputs of the system is also addressed.

2.2.1 Agent Technology

2.2.1.1 What is an agent?

An agent can be described as a piece of software that is capable of acting on behalf of others to accomplish a task. A more general description would state that an agent is a programme authorized with some degree of autonomy to represent others in the same manner they would represent themselves. Consider for instance a role undertaken by a Real Estate Agent, the main representative role of this agent is that it acts on behalf of the actual owner of a property in order to achieve some delegated tasks. This characteristic can be considered as the first

fundamental property of an agent. A second fundamental property is that they are autonomous, for instance, estate agents can arrange a viewing appointment for vacant properties without reference to the owners. A third property of an agent is the degree of proactivity and/or reactivity in their behaviour. An estate agent for instance will attempt to rent or sell a property despite any prior failed attempts, and it reacts to the changes may occur in the environment by modifying its plan accordingly. Agents have many different properties that they may possess in various combinations; these properties and their definitions can be found in appendix A.

Agents are normally classified according to the type of properties they enjoy. Different agents exist in literature such as software agents, intelligent agents, distributed agents, Information agents, collaborative agents, and mobile agents etc. Other forms of agents that do not possess the properties listed above are facilitator agents, broker agents, and manager agents, and so on. For instance, a broker agent coordinates the access and retains information about other agents. These forms can be more easily thought as roles that an agent can play—rather than the fundamental approach designed into an agent [112]. Software and broker agents are the main focus in the research that have been used to establish the monitoring and classification agency.

2.2.1.2 Software Agents

Software agents are among the most rapidly growing areas of research and development in Information Technology (IT). They can be found in a wide range of applications such as operating systems interfaces, workflow management, electronic commerce, web searches, and distributed problem solving. Despite this proliferation, there is no yet commonly agreed definition on what it is that makes a software entity an agent – [143] define it as “a persistent software entity dedicated to a specific purpose”; [136] takes agents to be “computer programs that simulate a human relationship by doing something that another person could do for you”; and [64] defines an agent as “a software entity to which tasks can be delegated”. A more specific definition of software agent that many agent researchers might find acceptable is: a software entity which functions continuously and autonomously in a particular environment, often inhabited by other agents and processes [142].

[103] Analysed a diverse list of resources seeking a common agreement of how agents differ from programs. They did find a number of key issues that permeate the research and

developer literature, namely: autonomous execution is central to agency; agents act autonomously to achieve a set of goals in the real world; agents require persistence to meet their goals; agents reason during the process of selecting the correct action to undertake; an intelligent agent acts on behalf of third party, with authority granted by this third party; agents interact with each other via an agent-communication language; agents engage in dialogue and negotiate and coordinate the transfer of information.

The Wooldridge definition is increasingly adopted amongst many, or probably fair to say most prominent researchers [117]. The definition states ‘*An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives*’, he also distinguishes between an agent and an intelligent agent, which is further required to be reactive, proactive and social [161], which is adapted from [159]. Software agents have a number of characteristics; according to [159], [69], and [117]; these properties are as the following:

- *Autonomous*: agents are independent and make their own decisions without a direct intervention of humans or other agents.
- *Situated*: agents exist in an environment to achieve a goal. This environment is often dynamic that change rapidly, which means that agents must respond to significant changes in the place where they reside.

Given that agents are often situated in dynamic environments, agents then need to be:

- *Reactive*: responding in a timely manner to changes in their environment.
- *Proactive*: agents pursue goals over time, that is, they are persistent; this is useful in that it makes agents more robust: an agent will continue to attempt to achieve a goal despite failed attempts.

These four properties distinguish agents from objects and other related software paradigms – such as object-oriented systems, distributed systems, and expert systems.

The central idea underlying software agents is that of delegation. The owner or user of software agent entrusts a task to the agent and the agent autonomously performs that task on behalf of the user. The agent must be able to communicate with the user, to receive instructions and provide the user with the results of its activities. Furthermore, agents should be able to interact with other software agents in order to complete their own problem solving

and to help others with their activities where appropriate. The agent must also be able to monitor the state of its execution environment and make the decisions necessary for it to carry out its delegated tasks [54].

(Note, from now on, the term agent will usually mean *software agent*).

2.2.1.3 What makes software agents distinctive?

Software agents can be used in a wide variety of applications. Developers use software agents to build intelligent systems in order to solve complex problems. They are autonomous; they have the capability to operate as a standalone process and perform action without user input or intervention. A software agent is communicative; it communicates with a user, other software agents, or other software processes. Software agents are more useful and powerful when they cooperate in an environment called an agency to achieve a task. Users can create software agents and distribute them in multiple remote hosts to perform monitoring of environments, collection and data retrieval, users then call the agents at some stage to return the collected data for analysis purposes.

Since the flow of control is not tied-up with the user when using software agents, the user does not need to have a permanent connection to the network until the monitoring and the collection tasks are done. In consequence, the network traffic is reduced, the server load is minimised and the user connection costs (to an Internet Server Connection – ISC) are cut-down enormously [147].

One of the differences between software agents and other techniques such as RPC (Remote Procedure Call) and Java RMI (Remote Method Invocation) is the flow of control from the user-side (Client) and the server. In RPC/RMI approaches, users always manage the control. In software agents approach users have limited control, but they lose it as the agents are distributed. Removing complete control from the user is not normally desired, in some cases, users can have some authority to control the state of an agent, to start, stop or pause an agent when required, but not the behaviour, for instance, the way agents monitor and collect data from an environment such as the Internet.

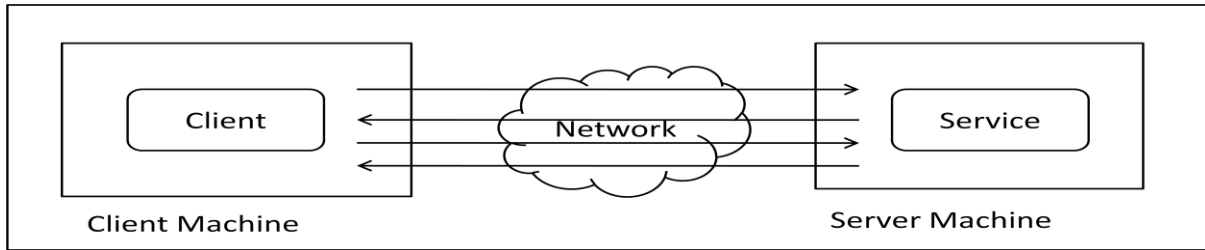


Figure 2.1 Traditional client-server approach

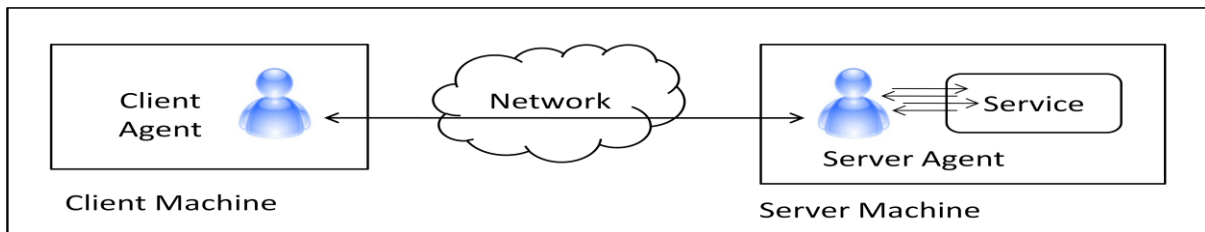


Figure 2.2 software agent based approach

Figures 2.1/2.2 represent the traditional client-server approach (RPC) versus the agent-based approach. In the first approach the user (client) have a direct and continuous access to the server. In the second approach, the user is free to engage with other tasks once the agents have been distributed, and the agents keep running in the background of the servers until they achieve their delegated tasks, when agents reach this point, they are called by the client agent to return the collected data for further processing.

The agent-base solution is useful and attractive because of the previously mentioned properties, such as being running autonomously, continuously, and asynchronously without user instructions. Agents are proactive and reactive to changes in their environment. The capability of communicating with other agents by exchanging synchronous/asynchronous messages makes software agents even more attractive. The system developer no longer has to design communication protocols and message format. This is the agents' responsibility which is provided as part of the agent mechanism. All the developer needs to do is to specify what the agents need to do in any given situation.

2.2.1.4 Agent Usage

[69] Outlines some of the key systems in which agent-based applications have been reported:

- *Industrial Applications* Industrial applications of agent technology were among the first to be developed, and today, agents are being applied in a wide range of industrial systems, such as manufacturing, process control, telecommunications, air traffic control, and transportation systems.
- *Commercial Applications* While industrial applications tend to be highly-complex, bespoke systems which operate in comparatively small niche areas, commercial applications, especially those concerned with information management (Information gathering and information retrieval), tend to be oriented much more towards the mass market. Other applications in this area include electronic commerce and Business Process Management.
- *Entertainment* leisure applications such as computer games can be extremely challenging and lucrative. Agents have an obvious role in computer games, interactive theatre, and related virtual reality applications: such systems tend to be full of semi-autonomous animated characters, which can naturally be implemented as agents.

Medical Applications: Medical informatics is a major growth area in computer science. Thus, it is not surprising that agents should be applied in this domain. For instance, Agent-based applications have been applied in the areas of patient monitoring and health care.

2.2.1.5 How Agents Communicate

One of the most important characteristics of agents is interaction. Agents interact to share information, negotiate, and exchange messages in order to perform tasks to achieve their goals. Without communication, different agents cannot know from each other who is doing what and how they can cooperate [17]. Therefore, communication is a must if we want to set up a useful multi-agent system [17]. There are several approaches in literature that show how this communication can be achieved. The most important approaches are communication using communication protocols and communication using an evolving language [17]. However, the two most well known communication languages are KQML (Knowledge Query Meta Language) [74] and, FIPA ACL (Agent Communication Language) [38].

XML (eXtensible Markup Language) is becoming widely adopted standard by a large number of software vendors in the areas of storing, describing information and for data

interchange on the internet. Its flexibility and ability to clearly represent and identify data makes XML ideal for transferring data between agents.

The KQML and FIPA ACL agent communication languages, the most well known, have been converted to simple XML form. Several XML-based schemas are being developed, such as Resource Description Framework (RDF) [127], XML-Data [164] and Document Content Definition (DCD) [29]. Most existing XML schemas focus on strong data formats. For instance, RDF focuses on how to represent semantic networks; XML-Data considers basic data types such as Integer, Long and Data; DCD is a simplification of RDF that takes account of the data types of XML-Data.

2.2.1.5.1 Agent Communication Languages

Many agent communication languages have been developed that can be used for coordination and communication between agents. The increase number of these languages made it difficult to handle one standardized language, so that different parties can build their agents to interoperate. [153] enumerates a list of such languages. Table 2.1 shows and describes the most prominent examples of ACLs.

Agent communication language	Description
KQML (“Knowledge Query and Manipulation Language”)	It is perhaps the most widely used agent communication language [153].
ARCOL (“ARTIMIS COmmunication Language”)	It is the communication language used in the ARTIMIS system [153]. ARCOL has a smaller set of communication primitives than KQML, but these can be composed [153].
FIPA-ACL (“FIPA Agent Communication Language”)	It is an agent communication language that is largely influenced by ARCOL [153]. Together FIPA-ACL, ARCOL, and KQML establish a quasi standard for agent communication languages [153].
KIF (“Knowledge Interchange Format”)	It is a logic-based language that has been designed to express any kind of knowledge and meta-knowledge [153]. KIF is a language for content communication, whereas languages like KQML, ARCOL, and FIPA-ACL are for intention communication [153].

COOL (“Domain independent COOrdination Language”)	It aims at explicitly representing and applying coordination knowledge for multi-agent systems and focuses on rule-based conversation management [153]. Languages like COOL can be thought of as supporting a coordination/communication (or “protocol-sensitive”) layer above intention communication [153].
---	---

Table 2.1 Most prominent Agent Communication Languages

Several other languages showing unique properties have been developed [153], for example, Internet Communication Language (ICL0, AgentTalk), Communication and Coordination Language (Cola), Tuple Centres Spread over Networks (TuCSon), LuCe, Simple Thread Language ++ (STL++), and Strictly Declarative Modelling Language (SDML).

2.2.1.5.2 Agent Transportation Mechanisms

Agents normally exchange messages when communicating with each other to perform tasks. These messages can be sent either synchronously or asynchronously.

The transportation mechanism should support unique addressing as well as role-based addresses [112]. Furthermore, the transportation mechanism must support unicast, multicast, and broadcast modes and such services as broadcast behaviour, non-repudiation of messages, and logging [112]. Several agent transportation mechanisms have been developed, for example, CORBA (“Common Object Request Broker Architecture”), OMG (“Object Management Group”) Messaging Services, RMI (“Remote Method Invocation”), DCOM (“Distributed Component Object Model”) and, Enterprise Java Beans Events.

2.2.1.6 Software Development Methodologies for Agent-based Systems

As agents are gaining acceptance as a technology and are being used, there is a growing need for practical methods for developing agent applications. However, one of the most fundamental obstacles to large-scale take-up of agent technology is the lack of mature software development methodologies for agent-based systems [94]. Numerous agent oriented methodologies have been proposed in the literature such as Prometheus [78, 79], GAIA [162], TROPOS [118, 34] and MaSE [134, 135], just to name a few. Additionally, there are other software development methodologies such as Agile, RUP, and Shlaer-Mellor that can

be considered as alternatives for the design of the agent system. The sub-section below briefly presents Agile, RUP, and Shlaer-Mellor and shows why Prometheus has been chosen for the design of the agent system.

2.2.1.6.1 Agile Methodologies

Agile software development was one of the attempts to provide organizations with lighter weight and faster software development processes. It is a group of software development methodologies based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing and cross-functional teams [26]. Agile is a people oriented methodology that can be used to increase productivity across the development lifecycle and reduce risks. Agile methodologies include Extreme Programming (XP), Scrum, Feature Driven Development (FDD), Dynamic System Development Method (DSDM), Adaptive Software Development (ASD), and Agile Manifesto. These methodologies have been compared in order to value one method over the other using three selected aspects: key points, special features, and identified weaknesses [7]. Key points detail the methods, principles, aspects, or solutions. Special feature describes one or several aspects of methods that differentiate them from others. Finally, identified weaknesses relate to some aspects of a method that have been documented in literature. Table 2.2 modified from [26, 40] summarizes the general features of Agile Methods [7].

Method Name	Key Points	Special features	Identified weakness
ASD	Adaptive culture, collaboration, mission-driven component based iterative development	Organizations are seen as adaptive systems. Creating an emergent order out of a web of interconnected individuals	ASD is more about concepts and culture than the software practice
DSDM	Application of controls to RAD, use of timeboxing and empowered DSDM teams.	First truly agile software development method, use of prototyping, several user roles : “ambassador”, “visionary” and “advisor”	While the method is available, only consortium members have access to white papers dealing with the actual use of the method
XP	Customer driven development, small teams, daily builds	Refactoring - the ongoing redesign of the system to improve its performance and responsiveness too change	While individual practices are suitable for many situations, overall view & management practices are given less attention
SCRUM	Independent, small, self-organizing development	Enforce a paradigm shift from the “defined and	While Scrum details in specific how to manage

	teams, 30-day release cycles.	repeatable” to the “new product development view of Scrum”	the 30-day release cycle, the integration and acceptance tests are not detailed
FDD	Five-step process, object-oriented component (i.e. feature) based development.	Method simplicity, design and implement the system by features, object modeling	FDD focuses only on design and implementation. Needs other supporting approaches.

Table 2.2 General Features of Agile Methods [Table modified from 26, 40]

2.2.1.6.2 The Rational Unified Process (RUP)

RUP is a Software Engineering Process. It provides a disciplined approach to assigning tasks and responsibilities within a development organization [125]. Its goal is to ensure the production of high-quality software that meets the needs of its end-users, within a predictable schedule and budget [62, 76]. RUP is a methodology that provides guidance to developers to use the Unified Modelling Language (UML) effectively. The *Unified Modeling Language (UML)* is the open method industry-standard language used for specifying, visualizing, modifying, constructing, and documenting the artifacts of an object-oriented software intensive system under development. [71].

The UML was originally created by Rational Software Corporation, a division of IBM since 2003, and is now maintained by the standards organization Object Management Group (OMG) [12]. Unlike Agile that provides basic guidelines for the development of software projects, RUP is largely used in the business community for the development of object-oriented projects using the UML modelling techniques.

2.2.1.6.3 Shlaer-Mellor Method

“The Shlaer-Mellor Method is a well-defined [139, 140, 141] and disciplined approach to the development of industrial-grade software. It is based on the object-oriented paradigm, and has developed over the past dozen years in the pragmatic environment of real-world projects” [101]. The Shlaer-Mellor Method is one of the most popular and widely used methods for the development of the Object-Oriented based projects. It was established in 1980 by Sally Shlaer and Stephen Mellor as a solution to overcome the weaknesses in the existing software development methodologies.

The Shlaer-Mellor Method comprises the following steps, grouped into two separate activities: Object-Oriented Analysis (OOA) and Recursive Design (RD) [101].

I. Object-Oriented Analysis

- Partition the system into domains.
- Perform a detailed Object-Oriented Analysis (OOA) on each domain that must be built. OOA consists of developing an Object Information Model (OIM), State Model (SM) and Process Model (PM) (see figure 1).
- Verify the analysis of each domain.

II. Recursive Design

- Determine an architecture and specify the rules for translation of the OOA models into implementation.
- Build the translation components.
- Translate the OOA models into implementation.

2.2.1.6.4 The Prometheus Methodology

Prometheus is intended to be a *practical* methodology. As such, it aims to be complete: providing everything that is needed (start-to-end process) to specify and design agent systems. Other distinguishing features of the Prometheus methodology are [80]:

- Prometheus is *detailed* – it provides detailed guidance on *how* to perform the various steps that form the process of Prometheus.
- Prometheus supports (though is not limited to) the design of agents that are based on goals and plans.
- Prometheus covers a range of activities from requirements specification through to detailed design.
- Prometheus allows the creation of the communication between the agents when a message is sent from one agent to another.
- The methodology is designed to facilitate tool support, and tool support exists in the form of the *Prometheus Design Tool* (PDT).

Prometheus, through its three phases (figure 2.3), system specification, architectural design, and detailed design, defines an in-depth process for specifying and designing agent-oriented systems. The process defines a range of artefacts some of which are used as permanent, and

others that are used as ‘stepping stones’ for other artefacts. Artefacts include goals, capabilities, events, plans and data structures, actions and percepts.

The system specification phase focuses on the identification of system goals, developing use case scenarios that demonstrate the operation of the system to be developed, determining the basic functionalities of the system and the specification of the interface between the system and its working environment by identifying the result actions.

The architectural design phase builds on the artefacts (deliverables) from the system specification phase to determine the composition of the agents the system will contain and how they will interact. This stage is also used to capture the system’s overall structure and its dynamic behaviour.

The detailed design is used to establish the internal design of each agent within the system and is not dependent on any one particular development platform/environment.

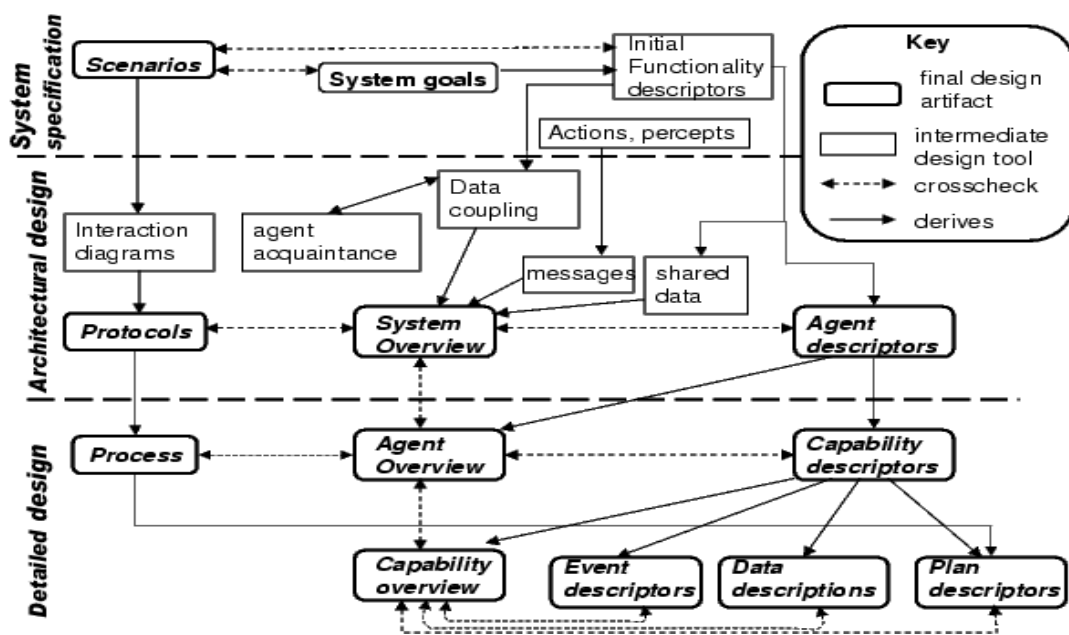


Figure 2.3, the phases of Prometheus methodology

Although Agile, RUP, and Shlaer-Mellor are general purpose software development methodologies, these methods could be used as alternative solutions for the design of the agent system. In contrast, Prometheus is developed specifically for specifying and designing agent-oriented systems, therefore it has been chosen to undertake this phase of the project. Additionally, Prometheus and its accompanying tool (PDT) beyond specifying and designing

the agents required to construct the agency specify also the communication needed between these agents in order to coordinate to achieve their delegated tasks. Furthermore, Prometheus has been compared to other existing software development methodologies for agent-based systems [72, 73]. The features of Prometheus distinguish it from these methodologies, but none of them have all of the Prometheus features described above [80].

2.2.1.7 Multi-Agent Systems

Research into systems composed of multiple agents was initially carried out under the banner of Distributed Artificial Intelligence (DAI). Traditionally, DAI has been divided into two main camps [11]: Distributed Problem Solving (DPS) and Multi-Agent Systems (MAS). More recently, the term “multi-agent systems” has come to have a more general meaning, and is now used to refer to all types of systems composed of multiple (semi-) autonomous components.

Distributed Problem Solving (DPS) is concerned with how a problem is solved by a team of multiple cooperating distributed agents, each agent having partial data and knowledge of the problem and the problem solving environment. In contrast, research in MAS is concerned with the study, behaviour, and construction of a collection of possibly pre-existing autonomous agents that interact with each other and their environments in order to achieve a common goal [144].

An MAS can be defined as a loosely coupled network of problem solvers that interact to solve problems that are beyond the individual capabilities or knowledge of each problem solver [77]. These problem solvers, often called agents and are autonomous and can also be heterogeneous in nature. [144] identifies four characteristics of MAS that (1) each agent has incomplete information or capabilities for solving the problem and, has a limited viewpoint; (2) there is no system global control; (3) data are decentralized; and (4) computation is asynchronous.

The increasing interest in MAS research includes their ability to [46, 69, 144]:

- Solve problems that are too large for a centralised single agent to do due to resource limitations or the sheer risk of having one centralised system.
- Allow for the interconnecting and interoperation of multiple existing legacy systems, e.g., expert systems, decision support systems, etc.;

- Provide solutions to inherently distributed problems, e.g., air traffic control [19].
- Provide solutions which draw from distributed information sources;
- Provide solutions where the expertise is distributed, e.g., in health care provisioning;
- To enhance:
 - Speed (if communication is kept minimal),
 - Reliability (capability to recover from the failure of individual components, with graceful degradation in performance).
 - Extensibility (capability to alter the number of processors applied to a problem), the ability to tolerate uncertain data and knowledge.
- To offer conceptual clarity and simplicity of design.

One of the important factors that are fostering MAS development is the increasing popularity of the internet, which provides the basis for an open environment where agents interact with each other to reach their individual or shared goals. To interact in such an environment, agents need to overcome two problems: they must be able to find each other; and they must be able to interact [129]. Once these problems have been overcome, agents can be engaged in multiple, parallel interactions with other agents. Here, agents can act as a society [110].

[124] states that the area of multi-agent system has grown into a promising technology offering sensible alternatives for the design of distributed, intelligent systems. Several efforts have been made by academic researchers, by industrialists, and by several standardisation consortia in order to provide new tools, methods, and frameworks so as to establish the necessary standards for a wide use of MAS as a technology of its own, not only as a new paradigm.

2.2.1.8 Compatibility in Multi-Agent Systems

For decades, Developers have been facing challenges constructing ever larger and more complex software applications that have the capability to deal with complex problems. These applications must run across multiple and different platforms and operating systems and inter-operate with other heterogeneous systems in order to achieve this goal. Such applications are difficult to produce using the traditional software technologies because of some limitations such as coping with dynamic environments that change rapidly. However, It would appear that agent-based technologies represent a promising tool for the deployment of such applications because they offer the high-level software abstractions needed to manage

complex applications and because they were invented to cope with distribution and interoperability [13, 41, 67, 83, 108, 159].

The sub-section below briefly discuss some of the most important efforts that define interoperability between agents on different types of platforms, with emphasis on FIPA efforts

2.2.1.8.1 Different Approaches of Standardizations

There are three important agent standardization efforts which are attempting to support interoperability between agents on different types of agent platform: KQML community, OMG's MASIF and FIPA [120, 121].

KQML (Knowledge Query Meta Language) [74] was one of the first initiatives to specify how to support the social interaction characteristic of agents using a protocol based on speech acts and is now also one of the most pervasive ACLs. KQML was developed at UMBC by [35] and has spread throughout the academic community. KQML however isn't a true de facto standard in the sense that there is no consensus in the community on a single specification (or set of specifications) and it has not yet been ratified by common agreement between members of an organization and forum of some standing in the community. As a result, variations of KQML exist such as KQML classic, KQML '93 and KQML-Lite, leading to different agent systems that speak different dialects and that are not able to interoperate fully.

OMG MASIF [88] the Object Management Group Mobile Agent System Interoperability Facility, MASIF, differs from both KQML and FIPA in that it regards the defining characteristic for an agent as its mobility of the agent from one location to another. MASIF does not support or standardize interoperability between non-mobile agents on different agent platforms. Further, MASIF restricts the interoperability of agents to agents developed on CORBA platforms whereas the focus of FIPA is to directly support the interoperability of agents deployed on agent frameworks which can support varied message transport protocols.

OMG is exploring how to support other types of software agents than mobile agents. It has issued a Request For Information (RFI) on agents in 1999 [113], and FIPA has supplied its specifications as input to this request.

FIPA [38] “is a non-profit international organization that is dedicated to promoting the industry of intelligent agents by openly developing specifications supporting interoperability among agents and agent-based applications”. It was officially accepted by the IEEE Computer Society as its eleventh standards committee on 8 June 2005. “The standardization work of FIPA is in the direction to allow an easy interoperability between agent systems, because FIPA beyond the agent communication language specifies also the key agents necessary for the management of an agent system, the ontology necessary for the interaction between systems, and it defines also the transport level of the protocols” [8]. The use of common communication language, such as KQML, is not enough to easily support interoperability between different agent systems. The core mission of the FIPA standards consortium is to facilitate the interworking of agents and agent systems across multiple vendors’ platforms [38].

In contrast to MASIF, “KQML and FIPA both define interaction in terms of an Agent Communication Language (ACL) whereas MASIF defines interaction in terms of Remote Procedure Calls (RPC) or Remote Method Invocation (RMI). In contrast to the traditional RPC-based paradigm, the ACL as defined by FIPA provides an attempt at a universal message-oriented communication language. The FIPA ACL describes a standard way to package messages, in such a way that it is clear to other compliant agents what the purpose of the communication is” [121].

Since FIPA was established the membership of companies and organizations has been increased. The current number of members for 2010-2011 is 54 members. The increase of the members incorporated into FIPA, the presence of companies such as IBM, NASA, Intel, Philips etc., and the utilisation in large-scale projects such as [2, 18, 33] based on FIPA’s specifications; are factors that are likely to contribute to making FIPA specifications as universally accepted standard.

2.2.1.8.2 Overview of FIPA Architecture

In 2002 FIPA published the standards describing the multi-agent systems [36, 37]. According to them the agents are programs that expose and consume services. The main components of the multi-agent system in this standard are shown in FIPA's Agent Management Reference Model (see figure. 2.4). Agent management provides the normative framework within which FIPA agents exist and operate. It establishes the logical reference model for the creation, registration, location, communication, migration and retirement of agents.

The entities contained in the reference model are logical capability sets (that is, services) and do not imply any physical configuration. Additionally, the implementation details of individual Agent Platforms (APs) and agents are the design choices of the individual agent system developers.

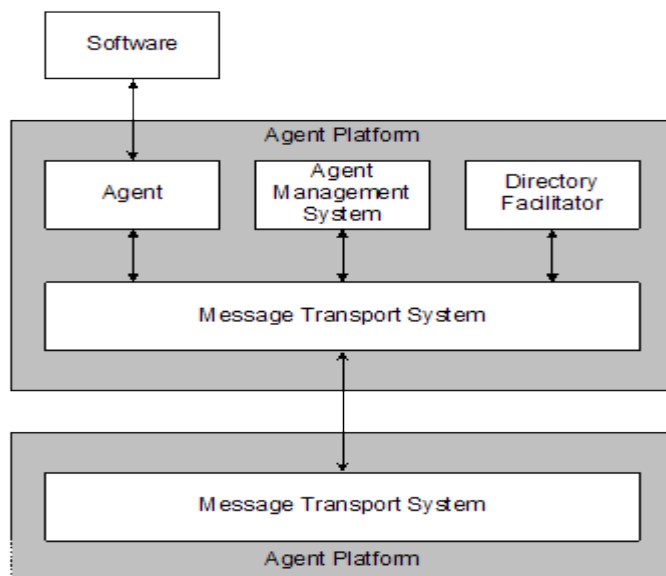


Figure 2.4 Agent Management Reference Model

An **Agent** is a computational process that implements the autonomous, communicating functionality of an application. Agents communicate using an Agent Communication Language. An Agent is the fundamental actor on an AP which combines one or more service capabilities, as published in a service description, into a unified and integrated execution model.

A **Directory Facilitator (DF)** is an optional component of the AP. It provides yellow pages services to other agents. Agents may register their services with the DF or query the DF to find out what services are offered by other agents. The information stored about the services exposed by the agents in the systems, are, for example, services' names, parameters and, type of returned values...etc. Multiple DFs may exist within an AP and may be federate.

An **Agent Management System (AMS)** is a mandatory component of the AP. It offers white pages services to other agents. The information about the agents such as agents' names and addresses are stored in this component. The AMS exerts supervisory control over access to and use of the AP. Only one AMS will exist in a single AP.

A **Message Transport Service (MTS)** is the default communication method between agents on different APs (see [39]).

An **Agent Platform (AP)** provides the physical infrastructure in which agents can be deployed. The AP consists of the machine(s), operating system, agent support software, FIPA agent management components (DF, AMS and MTS) and agents.

It should be noted that the concept of an AP does not mean that all agents resident on an AP have to be co-located on the same host computer. FIPA envisages a variety of different APs from single processes containing lightweight agent threads, to fully distributed APs built around proprietary or open middleware standards.

Software describes all non-agent, executable collections of instructions accessible through an agent. Agents may access software, for example, to add new services, acquire new communications protocols, acquire new security protocols/algorithms, acquire new negotiation protocols, access tools which support migration, etc.

2.2.2 .NET Framework

The .NET Framework is a key part of Microsoft's .NET software development platform [148]. .Net Framework is a development environment for building, deploying, and running Web Services and Web Applications that are accessible through client machines from across the globe [152]. .NET framework consists of two main parts: the common language runtime (CLR) and .Net Framework class library [1]. These two components provide the execution engine and programming APIs for building .NET applications. The concept of CLR is significant in making .Net Framework, platform and language independent. The class library

includes predefined sets of functionality that developers can use in their own applications, including ASP.NET for Web applications and Web services, Windows Forms for smart client applications, Windows services for developing long-running executable and autonomous processes, and Windows Communication Foundation (WCF) services for developing distributed applications. Visual Studio.NET is also a key part of Microsoft's developer platform and is a complete set of development tools that can be used to target any of the .NET features.

This chapter describes some of the .NET Framework techniques that have been used for the development of the Multi-Agent classification system, namely the Windows Services for the development of the software agents and, the WCF services for the distribution of the agents.

2.2.2.1 Agent Based Windows Service

An agent based on the concept of a .NET Windows Service has been explored by [126, 55, 56]. Windows services enable developers to create long-running executable applications that run in their own Windows sessions [61]. The services typically are configured to start automatically when the operating system loads into memory, but can also be paused, stopped, and restarted manually, programmatically using the Microsoft Management Console (MMC) utility, or using a third party utility [61]. This concept has been used in the construction of the data gathering and classification system that is configured to start automatically at user log-on to continuously monitor and analyse end user activities developing applications in the spreadsheet domain.

A Windows Service is designed to not require user intervention and to run unattended and autonomously. To launch a Windows service it must first be installed on the host machine and then the service will be started as a process when the computer boots. Visual Studio ships components to install service applications as this type of application by itself is not capable of being installed.

When building the service the author is required to add an installer class. This class hosts two objects named `ServiceProcessInstaller1` and `ServiceInstaller1`. The Service Installer object determines via its properties how the service will behave during operation. The `startType` property as its name suggests dictates the manner in which the service is started. Three options are available to the developer automatic, manual and disabled. The

ServiceProcess Installer object determines the service's security context and contains username and password properties.

After installers are added to the application, the next step is to create a setup project that will install the compiled project files and run the installers needed to install the service. In order to create a complete setup project, the service project's output must be added to the setup project and then a custom action must be added to have the service installed. Another method of installation is via using the installUtil utility from the command prompt.

2.2.2.2. .NET Windows Communication Foundation

Windows Communication Foundation (WCF) is Microsoft's distributed messaging technology that ships as part of the .NET Framework 3.0 for building connected, service-oriented applications. It enables developers to build secure, reliable, transacted solutions that integrate across platforms and interoperate with existing investments [157]. WCF unifies and extends the functionality of existing programming paradigms that have been previously used on the windows platform to achieve similar purposes, namely .NET Remoting, Enterprise Services (ES), ASP.NET Web Services (ASMX), and Web Services Enhancements (WSE) [95].

WCF is designed in accordance with SOA principles to support distributed computing where services are consumed by consumers. Clients can consume multiple services and services can be consumed by multiple clients. Client applications can directly consume services through Endpoints exposed by the service. Endpoints are locations through which messages can be sent or received.

WCF client does not interact with the service directly, even when dealing with a local, in-memory service. Instead, the client always uses a proxy to forward the call to the service. WCF allows the client to communicate with the service across all execution boundaries. On the same computer, the client can consume services in the same application domain, across application domains in the same process, or across processes [95]. Across computer boundaries, the client can interact with services in its intranet or across the Internet (see figure 2.5). Clients and services interact by sending and receiving SOAP messages. WCF services can be accessed over a variety of supported protocols, including HTTP, TCP, Named pipes, and MSMQ.

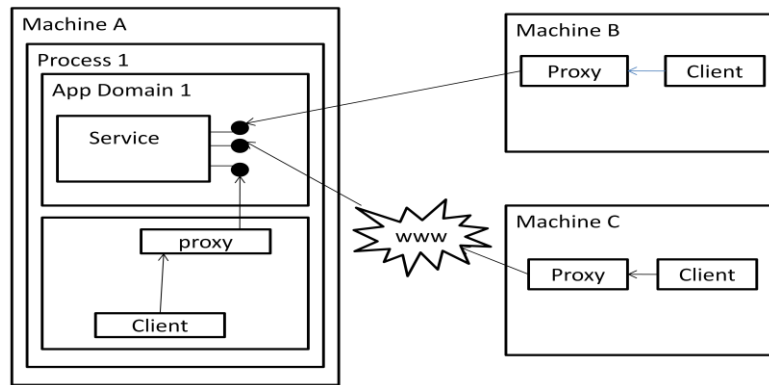


Figure 2.5 Same/cross-machine communication using WCF

A WCF service has three primary components (figure 2.6):

- Service class implemented in C# or Visual Basic or another CLR-based language that implements one or more services to be provided.
- Host environment in which the service runs.
- One or more endpoints that allow clients to access the service. All communications with a WCF service happens via the service's endpoints using SOAP messages.

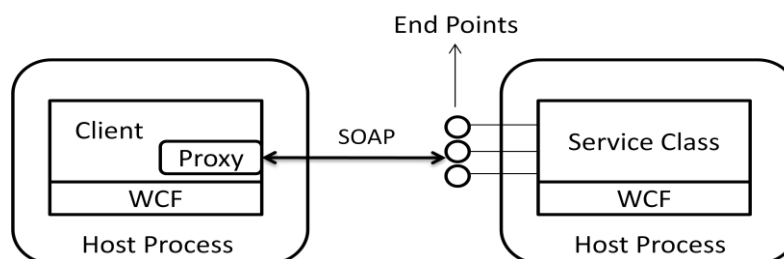


Figure 2.6 WCF primary components

Every endpoint consists of three important things (figure 2.7) where, what and how:

Address (Where): indicating where this endpoint can be found. Addresses are URLs that identify a machine and a particular endpoint on that machine.

Binding (How): determining how this endpoint can be accessed. WCF includes predefined bindings for most common communication protocols such as SOAP over HTTP, SOAP over TCP, and SOAP over Message Queues etc.

Contract (What): It defines an agreement or protocol how client should communicate with your service. Technically, it describes parameters and return values for a method.

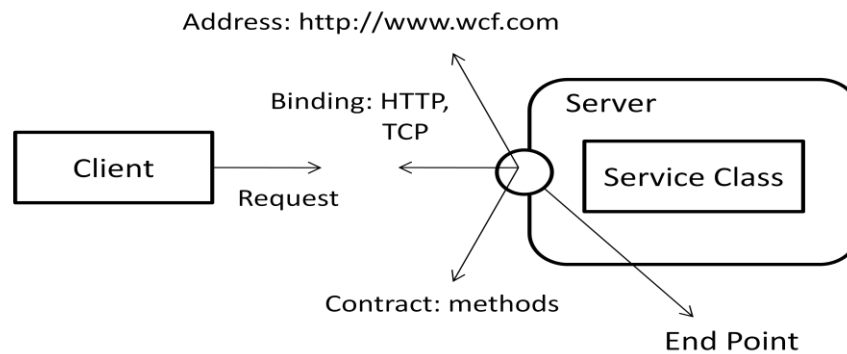


Figure 2.7 End Point Architecture

2.2.2.3 Hosting WCF Services in a Windows Service Application

Since the project relies on a service-oriented architecture, creating robust services was one of the top priorities. The most important driver behind the robustness was where/how the services can be hosted. WCF Services can be hosted in Internet Information Services (IIS); in Windows Activation Service (WAS); or in any managed application process including console-based applications, Windows Forms, Windows Presentation Foundation (WPF), or managed Windows Service applications. Hosting in windows service applications is a logical and the selected choice, as they provide a process model particularly suited to applications that must live in a long-running executable and do not interact directly with the user and therefore do not display any form of user interface. This ends up tying the lifetime of the WCF services to the lifetime of the Windows Services.

Hosting WCF in a Windows Service application comes with a number of benefits:

“Automatic starting: The Windows Service Control Manager allows the developer to set the start up type to automatic, so that as soon as Windows starts, the service will be started, without an interactive logon on the machine” [22].

“Recovery: The Windows Service Control Manager has built-in support to restart services when failures occur” [22].

“Security identity: The Windows Service Control Manager allows the developers to choose a specific security identity under which they want the service to run including built-in system or network service accounts” [22].

“Manageability: In general, Windows operators know a lot about the Service Control Manager and other management tools that can work with Windows service installation and

configuration. This will improve the acceptance of Windows services in production environments; however, to make services maintainable, developers would probably have to add some instrumentation and logging features” [22].

“Support for all bindings and transports: Self-hosting doesn't limit developers in using any of the out-of-the-box bindings and transports whatsoever. On Windows XP and Windows Server 2003, IIS is limited to HTTP only” [22].

2.2.2.4 Justification of the .NET WCF Technology as the Distribution Mechanism

In the development of the distributed applications, performance, interoperability, security, reliability, and availability are some of essential quality attributes. They are important for the provider and consumer for the selection and evaluation of the services [60, 109].

Performance is one of the primary reasons for moving from the other distributed communication technologies to WCF. When migration distributed application developed using ASP.NET Web Services, Web Services Enhancements (WSE), .NET Enterprise Services (ES), and .NET Remoting to WCF, the performance is significantly better for WCF over the other existing technologies [98]. Literature contains few studies that compare the performance of the WCF services with other distributed technologies [47, 96, 97, 98]. Table 2.3 summarizes the results of a high-level performance comparison between WCF and existing Microsoft .NET distributed technologies.

Other Distributed Technologies	WCF Performance Advantage
ASP.NET Web Services	25%-50% faster
.NET Remoting	25% faster
.NET Enterprise Services	Dependent on load balancing 100 % faster (in some cases) Nearly 25% slower (in other scenarios)
WSE 2.0/3.0 implementations	400% faster

Table 2.3 High-level performance comparison between WCF and existing Microsoft .NET distributed technologies.

Security is an increasingly important issue in the development of distributed applications [70, 109, 137], this is due to increasing the demand on the internet and the expansion of business networking. Thus, it is important that distributed technologies are capable of building secured

interconnected applications. The security is not fully guaranteed when the response time becomes too high and the service is unavailable for the many periods [20]. Furthermore, meeting security standards is an important step toward achieving the best interoperability.

Increased interoperability is the most prominent benefit of SOA, especially when Web Services technology is considered [63]. Various languages and platforms have been used for developing distributed systems that support interoperability, such as the Common Object Request Broker Architecture (CORBA), Remote Method Invocation (RMI), Distributed Component Object Model (DCOM), and Remote Procedure Call (RPC). CORBA for example, is a standard for distributed computing and interoperability. CORBA provides the mechanism of implementation and platform transparency [87]. Commercial CORBA implementations are expensive which led to limiting the acceptance of this platform. Furthermore, the platform had a steep learning curve and was complex and hard to use correctly [52]. This technology also has insufficient features; it provides a rich functionality but fails to provide the features of security and versioning [52].

WCF is a distributed communication technology that provides distributed computing abilities, broad interoperability, and is a pure SOA platform. Prior to Microsoft's .NET WCF services, it was not an easy challenge to choose a particular technology to develop a distributed system due to the number of technologies that Microsoft provides. For example, developers could have used Web Services to develop systems that support interoperable machine-to-machine interaction in which Java based applications and .NET applications are operating; WSE to build distributed applications based upon the Web Service specifications quickly and easily. Developers could have used ESs to build distributed applications that can be deployed in an enterprise environment; MSMQ to enable communication across heterogeneous networks and across computers where connection is not always available. .NET Remoting builds high performance remote objects based applications over a network easily. All these technologies have their own advantages and disadvantages.

However, WCF was developed to take advantage of all the mentioned technologies via combining their capabilities in a unified manner into a single, service-oriented model for communications, thus WCF is the successor of all these technologies. Furthermore, WCF is based on open web service standards like SOAP, XML, and the Latest WS-* industry standards [91, 98, 137]. WCF is interoperable with native as well as non Microsoft technologies (e.g. Java) that meet these standards [91, 137]. These technologies including

COM, DCOM, RMI, MSMQ, and Web Sphere MQ works well in a particular scenario but not well in some other cases. On the other side, WCF works well in any scenario in which a Microsoft .NET assembly communicates with any other software entity [91].

2.2.3 Data Mining

Databases in the early 21st century contain data that can be measured in terabytes; these databases are rich with hidden information [49]. Data mining (sometimes called Knowledge Discovery) is the process of extracting useful information and insight from large amounts of data; this data usually has to be collected from various resources. Data mining creates models to find hidden patterns in large, complex collections of data, patterns that sometimes elude traditional statistical approaches to analysis because of the large numbers of attributes, the complexity of patterns, or the difficulty in performing the analysis [85].

There are many different approaches and data mining algorithms, each with its strengths and weaknesses. In December 2006 the IEEE International Conference on Data Mining (ICDM) identified the top 10 data mining algorithms: C4.5, K-Means, Support Vector Machine (SVM), Apriori, Expectation-Maximization (EM), PageRank, AdaBoost, K-Nearest Neighbours (KNN), Naive Bayes, and Classification and Regression Trees (CART). These top 10 algorithms are among the most influential data mining algorithms in the research community. The 10 algorithms cover classification, clustering, statistical learning, association analysis, and link mining, which are all among the most important topics in data mining research and development [163].

Data mining techniques can be divided into two different types of learning models, supervised and unsupervised models.

In supervised, or predictive, directed, or targeted modelling, the goal is to predict an event or estimate the values of a contentious numeric attribute [150]. In this type of models there are both the input attribute values in which the target value is known and the output value or “Target Attribute”. The input attributes or the predictors are the data from which the algorithm learns or “trains” about relationships between the predictor variables in order to generate the output or the target attribute (covered in more detail in chapter six). Target attributes are either binary attributes indicating (Yes/No) decisions or multiclass targets where the model predicts one of several target values for each case. Classification and estimation are examples of supervised models.

In unsupervised or undirected models there is no output field or “Target Attribute” to predict or classify, just input fields are involved [150]. In this type of modelling there is no learning from cases where the outcome variable (target value) is known. The goal is establish or

discover relationships between the attributes. Clustering techniques, data detection methods, and association rules are all unsupervised learning models.

2.2.3.1 Mining with Oracle Data Mining

Oracle Data Mining (ODM) is powerful data mining software embeds data mining within Oracle database. Oracle Data Mining helps find patterns in data, identify key attributes, discover new clusters and associations, and discover valuable new insights [115]. With Oracle Data Mining, everything occurs in the Oracle Database in a single, secure, scalable, platform for business intelligence. Traditional alternatives force you to extract the data out of the database to separate, unsecured and costly dedicated statistical, analytical or mining servers [115]. The ultimate goal is to improve businesses or find solutions to business problems, all centred on helping to predict, understand, and develop new insights of individual activities. Oracle Data Mining supports the following data mining functions [85]:

- Supervised data mining:
 - Classification: Grouping items into discrete classes and predicting which class an item belongs to
 - Regression: Approximating and forecasting continuous values
 - Attribute Importance: Identifying the attributes that are most important in predicting results
 - Anomaly Detection: Identifying items that do not satisfy the characteristics of "normal" data (outliers)
- Unsupervised data mining:
 - Clustering: Finding natural groupings in the data
 - Association Rules: Analyzing "market baskets"
 - Feature Extraction: Creating new attributes (features) as a combination of the original attributes

2.2.3.2 Classification Algorithms

Classification is a data mining function that assigns items in a collection to target categories or classes. The goal of classification is to accurately predict the target class for each case in the data [86]. Oracle data mining provides solutions to both binary classification problems

where the model predicts one of two target values for each case (0 or 1; Yes or No decisions) and multiclass classification problems where the model predicts one of several target values for each case, For example, a classification model could be used to classify Excel users into Excel Developer, VBA Developer and, Professional Excel Developer.

Oracle Data Mining supports four different classification algorithms for solving classification problems, Naive Bayes, Decision Tree, Adaptive Bayes Networks, and Support Vector Machine. The nature of data normally determines the right algorithm that provides the best solution to the problem.

2.2.3.2.1 Naïve Bayes

Naive Bayes looks at the historical data and calculates conditional probabilities for the target values by observing the frequency of attribute values and of combinations of attribute values [86]. Naïve Bayes can be used to solve both binary and multiclass classification problems.

This algorithm is based on Bayes theorem, a statistical principle for combining prior knowledge of the classes with new evidence gathered from data [146]. Studies comparing classification algorithms have found a simple Bayesian classifier known as the Naïve Bayesian classifier to be comparable in performance with decision tree and selected neural network classifiers. Bayesian classifiers have also exhibited high accuracy and speed when applied to large databases [50].

The Bayes theorem shown below states that the probability of event A occurring given that event B has occurred is proportional to the probability of event B occurring given that event A has occurred multiplied by the probability of event A occurring [85].

$$\text{Prob (B given A)} = \text{Prob (A and B)} / \text{Prob (A)}$$

Suppose A represents “the user is an Excel developer” and B represents “the user increases knowledge working on Excel application”, how it can be determined the likelihood that the Excel developer will increase knowledge.

To calculate the probability that an Excel developer increases knowledge given that the user is an excel developer, the algorithm counts the number of cases where A and B occur together as a percentage of all cases and divides it by the number of cases where A occurs alone as a percentage of all cases.

2.2.3.2.2 Decision Trees

The decision tree algorithm can be used for both binary and multiclass classification problems. However, it is like the Naïve Bayes, as it is based on conditional probabilities, but it differs that Decision Trees generate rules. These rules provide model transparency so that a business user, marketing analyst, or business analyst can understand the basis of the model's predictions, and therefore, be comfortable acting on them and explaining them to others [85]. In addition to transparency, decision trees algorithm produces accurate models, with good speed, and low cost.

The decision trees algorithm manages its own data optimization internally. The algorithm must repeatedly find the most efficient way to manage the split of a data set into smaller subsets (child nodes) as the tree being built.

Figure 2.8 is a typical decision tree example. It represents the concept *buys_computer*, that is, it predicts whether a customer at *AllElectronics* is likely to purchase a computer. In this example, each internal node (noleaf node) represents a test on an attribute. Each branch represents an outcome of the test. Each leaf node (or terminal node) represents a class (either *buys_computer* = yes or *buys_computer* = no), these leaf nodes are the actual classifications [50].

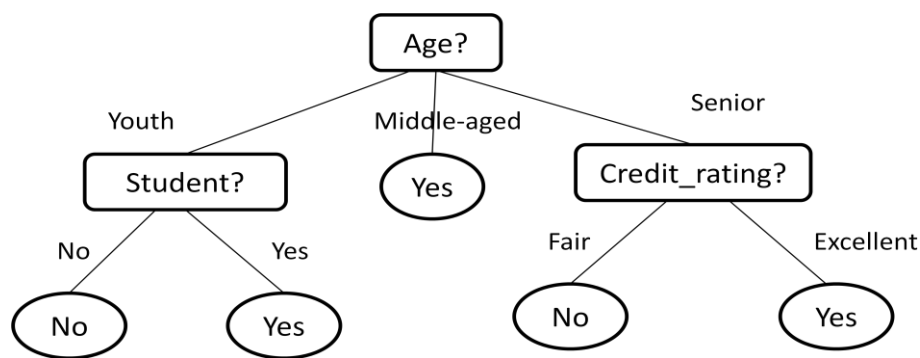


Figure 2.8 typical decision trees for the concept *buys_computer*

Decision tree algorithms can produce both binary trees where each internal node can be subdivided into two other child nodes, and can produce nonbinary trees.

2.2.3.2.3 Adaptive Bayes Network

Adaptive Bayes Network (ABN) is an Oracle proprietary algorithm that provides a fast, scalable, non-parametric means of extracting predictive information from data with respect to a target attribute [85]. However, it is like the decision trees algorithm as it generates rules, these rules are one of the advantages it provides over Naive Bayes. ABN rules provide model transparency so that a business user, marketer, or business analyst can understand the basis of the model's predictions and therefore, be comfortable acting on them and explaining them to others [85]. Performance and scalability are two other features can be provided by this algorithm. ABN can be used to solve both binary and multiclass classification problems.

ABN supports three different choices for Model Types, Single-Feature, Naive Bayes, and Multi-Feature.

If the default single-feature type is chosen, the algorithm starts with ranking the attribute values using the Attribute Importance feature that ODM provides. The use of this feature enables users to select the attributes that are most relevant and have positive influence on building the model process. Naive Bayes is used in this type as a baseline for building the model, and the number of attributes (Naive Bayes predictors) involved in the build process taken in order from the ranked list.

If the Naive Bayes is chosen as the model type, then the build process can be stopped at this point. The Attribute Importance features can be used here to measure the importance of the attributes that are most relevant to the building process, the resulting is a more efficient and accurate Naive Bayes model.

If Multi-Feature is chosen as a model type, the algorithm starts with building multiple network features, these features are like individual decision trees, every feature consists of at least one attribute (the root attribute) and number of levels. All features provide predictions of the target class probabilities. The depth of each network feature in the model is restricted by the Maximum Network Feature Depth parameter. Building multiple network features in this way improves the built model with each feature, resulting in a more efficient model.

During the building process, the model created at each chosen step is tested against the model created prior to the last step. In particular, when a network feature is built, it is tested against the model excluding that feature, and if the new feature does not make any improvement, it is discarded. When the number of consecutive discarded features reaches a number set

internally by the algorithm, the Adaptive Bayes Network stops building and what remains is the completed model [31].

2.2.3.2.4 Support Vector Machine

Support Vector Machine (SVM) is classification and regression prediction tool [6]. However, since the main focus in this subchapter is on Oracle classification techniques, thus Oracle data mining supports the classification of both linear and non-linear data. SVM classification is based on the concept of decision planes that define decision boundaries [86]. SVM attempts to separate the records that construct a data set into smaller subsets with identical target values (classes). Data records are separated by hyperplanes (decision boundaries) in the linear case, and in the case of non-linear data, cases are separated by non-linear separators.

[50] explained the mystery of SVMs using the simplest case of a two-class problem where the classes are linearly separable. They assumed a data set D given as $(X1, y1), (X2, y2), \dots, (X_{|D|}, y_{|D|})$, where X_i is the set of training tuples with associated class labels, y_i . Each y_i can take one of two values, either +1 or -1, corresponding to classes *buys_computer = yes* and *buys_computer = no*, respectively. They used an example based on two input attributes $A1$ and $A2$ as shown in figure 2.9. It is obvious from the graph that data are linearly separable, as a straight line can be drawn to separate all cases with class +1 from all of the cases of class -1.

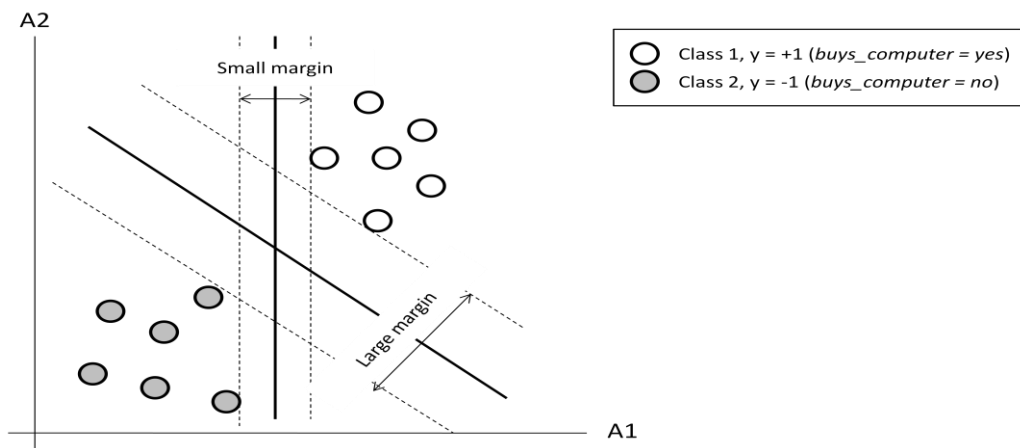


Figure 2.9 Linearly separable training data

An infinite number of separating lines (hyperplanes) can be drawn, and all the hyperplanes can correctly classify the given data tuples, but the question is, which hyperplane is the best

that will have the minimum classification error on previously unseen cases. SVM solves this problem by searching for the hyperplane with the largest margin between tuples, that is, the *Maximum Marginal Hyperplane*, as the hyperplane with the largest margin is expected to be more accurate at classifying future data tuples than the hyperplane with smaller margin. This case can be generalized to n dimensions, as ODM SVM classification supports both binary and multi-class classification problems.

The SVMs linear approach can be extended to create non-linear SVMs for the classification of linearly inseparable data [50]. This is achieved via transforming the input data into a higher dimensional space, where classes are linearly separable. Again, several hyperplanes can be drawn to linearly separate the classes. The SVM searches for the hyperplane that minimizes the classification error. This is the hyperplane that has the largest margin value that is capable of providing the most accurate classification to the new data.

2.2.3.3 Clustering Algorithms

Clustering algorithms partition data objects (patterns, entities, instances, observances, units) into a certain number of clusters (groups, subsets, or categories) [149]. Clustering is also known as unsupervised classification. Unlike classification and predictions, which analyse class-labeled data objects, clustering analyse objects without consulting a known class-label [50]. The class labels are not assigned to the cases (records) in the training data set as it happens in the supervised training simply because clustering do not use target classes. Such classes can be produced by the clustering models.

The objects are clustered or grouped based on the principle of maximizing the intraclass similarity and minimizing the interclass similarity [50]. The objects are clustered in a way that are similar within the same cluster and are dissimilar to the objects belonging to the other clusters. Figure 2.10 shows the grouping of the below coloured data items into three clusters.

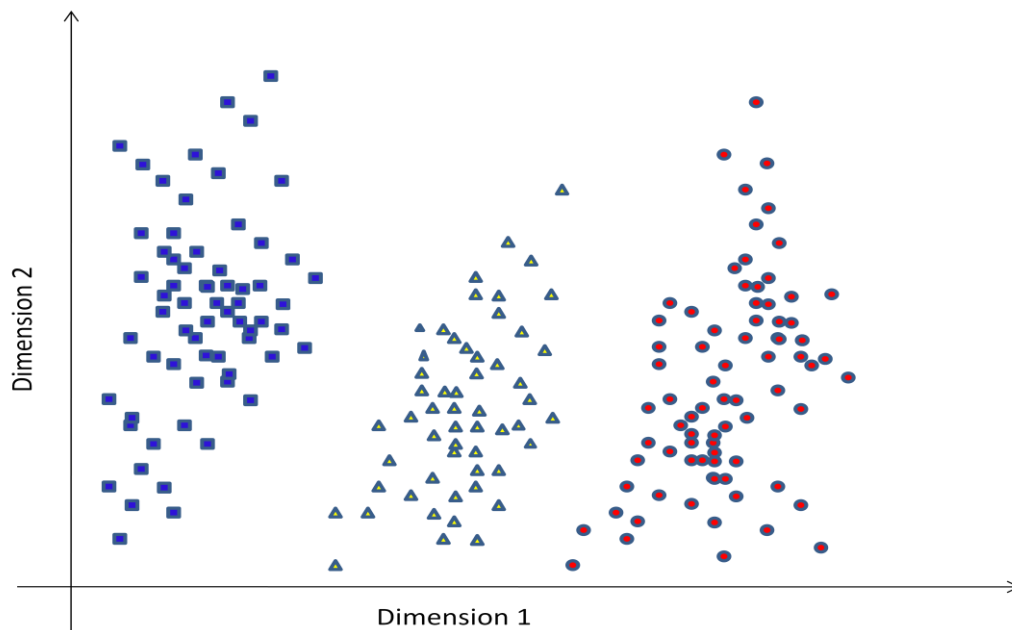


Figure 2.10 grouping of data items into three clusters

Clustering algorithms are used in many applications, including data mining, machine learning, image processing, marketing, and social networks. A number of different probabilistic clustering methods have been proposed in the literature, [43] splits them into two broad categories:

- Parametric Approaches, which includes the generative Models and Probability-based Models, the Gaussian Mixture model, C-Means Fuzzy Clustering, Reconstructive Models, K-Means and K-Median, and Deterministic Annealing.
- Non-Parametric Approaches, this includes the Hierarchical Clustering algorithms.

[100] categorized the clustering approaches further into:

- Partitional: Create an initial partition and then use an iterative control strategy to optimize objective.
- Hierarchical: Create a hierarchical decomposition (dendrogram) of the set of data (or objects) using more termination criterion.
- Density-based: Use connectivity and density functions.
- Grid-based: Create multiple-level granular structure, by quantizing the feature space in terms of finite cells.

“Clustering, when used for data mining is required to be [100]:

- Scalable
- Able to deal with different types of attributes
- Able to discover clusters with arbitrary shape
- Having minimal requirements for domain knowledge to determine input parameters
- Able to deal with noise and outliers
- Insensitive to order of input records
- High dimensionality
- Interpretable and usable”.

K-means algorithm for example, can be used to solve classification problems, when the target values are provided. This algorithm identifies the clusters with the same target value, and predictions of the new data (data of interest) are made by searching for the data items that are of the same type as the nearest cluster centre. Clustering algorithms can be applied to solve the classification problem of the project via grouping/dividing the data set into smaller data sets of the same similarity. This mechanism can possibly generate the different categories of Excel spreadsheet developers explained in table 2.6.

2.3 Related Work

The theoretical part of this dissertation has been covered by the previous sections. The reminder of this chapter is divided into two main sub-sections 2.3.1, 2.3.2, where each one provides a review of the most important efforts in the fields of end user computing, end user classification, from which the research of Multi-Agent Classification System has been influenced.

2.3.1 End User Computing

The Institute of End User Computing (IEUC) describes EUC as follows: ‘End User Computing embraces a broad range of technologies and techniques and looks of how they can be integrated and refactored to enhance system stability, security and flexibility’. End User computing is about empowering people to derive the greatest benefit from their computer with the least stress, [154]. The IEUC also states that there is no stereotypical end user with an array of technical abilities ranging from novice to expert and those areas of substantive interest vary greatly, [154].

The End User Computing area continues to grow by leaps and bounds as more and more users take control of their computer needs [84]. EUC technology has played an increasingly important role in assisting managers to make crucial decisions efficiently and effectively. The EUC technology is significantly different from the traditional computing systems in that while users of EUC technology can directly interact with the application software to enter information or prepare output reports, those of traditional computer systems had to interact with computers only through an analyst or programmer [30]. In 1992, [5] definition centres on end users as application developers.

2.3.2 End User Classification

A number of attempts have been made to classify end users into different groups based on meaningful criteria. The sub-section below briefly discusses the most important efforts; with emphasis to excel developer classification.

2.3.2.1 Different Approaches of End Users Classification

One of the earliest attempts to understand and classify the EUC phenomenon was the Codasyl report on end-user computing facilities [23]. This report divided end users into three categories: indirect end users, who use computers through other people; intermediate end users, who specify information requirements for reports they ultimately receive; and direct end users, who actually use terminals [130].

Contemporaneous with the Codasyl committee's report, McLean [90] proposed a somewhat broader classification of end users. He subdivided end users into:

- Data processing professionals (DPP), who wrote code for use by others
- DP users (DPU), who were further subdivided into DP amateurs (DPA) who wrote code for their own use and non-DP trained users (NTUs) who used code written by others.

[130] built upon the classifications of end-users proposed by Codasyl and McLean [23, 90].

They proposed a framework to classify end users consisting of six categories, namely:

- Non-programming end users
- Command level users
- End user programmers
- Functional support personnel
- End-user computing support personnel
- DP programmers.

This framework has a lack of a control dimension which could leave a negative impact on the classification of end users, and hence hinder its ability to devise strategies for improved management of end user activities within a modern organization.

[44] Strongly recommended the user cube framework developed by [25] that includes the control dimension. This framework consists of three dimensions, which are end-user development, end-user operation, and end-user control of Computer Based Information System (CBIS). Figure 2.11, provides a convenient way of visualizing these dimensions, the x-axis is operation, the y-axis is development, and the z-axis is the control of resources. The eight corners of the cube represent the extremes of each dimension. Corner (1,0,0) for example, represents the case of an individual or organization unit that is responsible for the operation of a system, but no development responsibility and control over resources. Similarly, corner (1,1,1) is totally responsible for the operation and development and has full

control over the resources of the system. An end user with a user cube classification of (0,0,0), or the end user who at best plays a consultative role in specifying the requirements [point on line (0,0,0)-(0,1,0), close to the origin (0,0,0)], are still end users, but would be examples of centralized computing environment.

In practice, we may find points such as (0.25, 1, 0.5) and (0.7, 0.3, 0.5), this means the user cube allows a classification not only on the surface (see table 2.4) of the cube, but also at any points within the cube. Thus the user cube provides a high level classification of end users that is not available within the [130] and other popular frameworks.

[25] suggested that without a definitive and consistent sub classification of end users within the organization there is a danger of contradictory and confusing advice that may have unforeseen implications for several categories of end users.

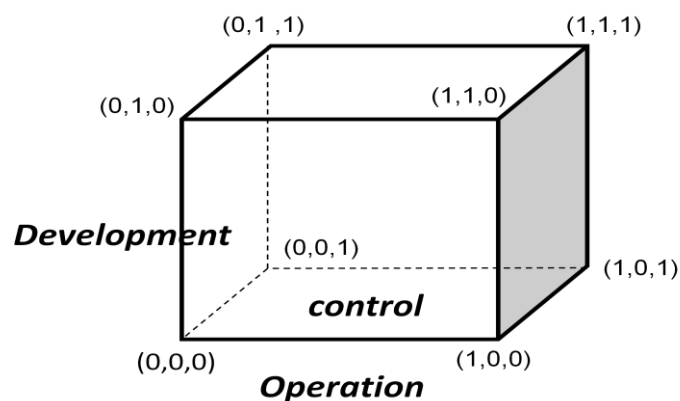


Figure 2.11, the User Cube

(x,y,z)	User Classification
(0,0,0)	User-consumer
(1,0,0)	User-operator
(0,1,0)	User-developer
(0,0,1)	User-controller
(1,1,0)	User-operator / developer
(0,1,1)	User-developer / controller
(1,0,1)	User-operator / controller
(1,1,1)	User-operator / developer / controller

Table 2.4 Types of End Users

2.3.2.2 End User Programmers Classification

Programming is defined by [106] as “behaviour in which sequences of procedural instructions... are written in a language that is compiled or interpreted,” then broadened her definition to include any attempt “to create an application that serves some functions”. However, it was reported in 1995, and widely disseminated in 2000 [10], that end user programmers would number 55 million in 2005, compared to fewer than 3 million professional programmers. These figures were extended with a fresh data and rich models recording the increase in computer usage rates among users as the number was already exceeded 64 million in 1997 and continues to grow [132]. Using survey results and projections from the Bureau of Labour Statistics (BLS), we estimated that over 90 million Americans will use a computer at work in 2012 (the year for which BLS published occupational projections), including 55 spreadsheet and database users and 13 million self-reported programmers, compared to fewer than 3 million professional programmers [132].

Programming and related concepts have been described in various ways, for example [132]:

- Programmers “utilize both command and procedural languages directly for their own personal information needs. They develop their own applications, some of which are used by other end-users” [130].
- “Development is the performance of any or all tasks of the system development process, whether traditional systems development life cycle or prototyping. It consists of the specification of system requirements, system design, programming, and/or system implementation and conversion” [25].
- ‘Program’ is defined as ‘a set of statements that can be submitted as a unit to some computer system and used to direct the behaviour of that system’ [28]. While the ability to compute ‘everything’ is not required, the system must include the ability to handle conditionals and iteration, at least implicitly” [104].
- “Writing high level, declarative, textual program specifications also constitutes programming, as does creating diagrammatic representations of system behavior. They demand the same basic activities and skills as conventional programming (even if the programmer is saved a considerable amount of time)... to create an application that serves some function” [106].

- “Programming may be defined as a procedure specification task by means of a computer language” [53].
- “End-user programming environments are quite diverse, including educational simulation builders, web authoring systems, multimedia authoring systems, e-mail filtering rules, CAD systems and... spreadsheets” [131], as well as other “special-purpose scripting languages” [155].
- “Setting a video cassette recorder to tape an upcoming TV show is not programming... Programming... is defined as the construction of a specification (sequence of instructions or program) for solving a problem by an agent other than the programmer” [24].
- “In a simple programming activity such as programming a VCR, the user is defining some abstract behaviour which is not directly observable because it will take place in the future... All computer users may now be regarded as programmers” [9].

The question here is how many end users can be categorized as programmers. A simple binary division of “end user programmers” from “end use non-programmers” provides inadequate insight into end user behaviour to guide future research and tool development [133]. Instead, [133] argued that end users exhibit a variety of practices ranging from programming-like to non-programming-like. They state that from this diversity they believe that they can characterise this distribution of behaviour on the basis of how end users represent abstractions (variables, functions, and data structures).

BLS software usage data from 2001 [132] a coarse-grained answer shown in table 2.5. it shows that only 15.2% out of 72,277 end users “do programming” at work, while 62.3% of the end users used spreadsheets and databases at work [133]. Furthermore, the BLS data do not explicitly address many other end user programming environments, such as “educational simulation builders”, web authoring systems, multi-media authoring systems, email-filtering rules, CAD systems” [131].

Question: Do You...	Thousands of Users	Percent of Computer Users
Use a computer at your main job?	72,277	<<100%>>
Connect to the Internet or use email?	51,895	71.8%
Do word processing or desktop publishing?	48,426	67.0

Use spreadsheets or databases?	45,029	62.3
Use a calendar or do scheduling on the computer?	38,235	52.9
Do graphics and design?	20,816	28.8
Do programming?	10,986	15.2

Table 2.5 Software application usage by US workers in 2001

The classification of end users has always been important aspect. Any classification scheme should provide the following:

- Support for the features of context, clearly defined categories, and testing in a relevant situation, [122].
- Support for abstractions as they provide a mechanism to characterise end user programming practices, [133].
- Support for the end user development activity indicators.
- A mapping from relevant existing classification schemes.

2.3.2.3 Spreadsheet Technology

End User Programmers (EUP) are people “who write programs, but not as their primary job function. Instead, they must write programs in support of achieving their main goal, which is something else, such as accounting, designing a web page, doing office work, scientific research, entertainment, etc. End user programmers generally use special-purpose languages such as spreadsheet languages or web authoring scripts, but some EUPs such as chemists, or other scientists, may learn to use ‘regular’ programming languages such as C or Java to achieve their programming goals”, [105].

The Microsoft Excel Spreadsheet provides an ideal environment for end users development and operation activities. This tool is installed on the desktops of four hundred million users [65]. [16] State that Excel consists of five fundamental concepts on which to create applications:

- Interface and Presentation – Worksheets and charts are used for data entry and reporting;
- Data Store – Worksheets store lists, tables etc. For application usage.

- Programming Environment – Visual Basic for Applications (VBA) provides Excel’s programming environment.
- Numeric Processing – The worksheet supports a declarative language for the purpose of numeric processing.
- Excel Object Model – Most of Excel’s functionality can be programmatically controlled from both inside and outside of the spreadsheet environment by use of supplied Dynamic Link Libraries (DLL).

2.3.2.3.1 Spreadsheet Complexity

Price, Waterhouse and Coopers (PWC), state that spreadsheets are a fundamental tool in the functional areas of financial reporting, analysis of management information, and tracking and monitoring of workflow to support operational processes. Spreadsheets vary greatly in their complexity and PWC differentiate this complexity with a three band categorization of low, moderate and high [123].

Low: This category consists of spreadsheets that serve to log data and information tracking systems.

Moderate: This category is typified by spreadsheets that perform simple calculations such as using formulae to total certain fields or calculate new values by multiplying two cells.

High: High complexity spreadsheets typically support complex calculations, valuations and modelling tools. These spreadsheets tend to use macros and multiple supporting spreadsheets where cells, values and individual spreadsheets are linked.

PWC identified a number of potential risks, errors and issues with spreadsheets:

- Complexity of the spreadsheet and calculations
- Purpose of the spreadsheet
- Number of spreadsheet users
- Type of potential input, logic and interface errors

- Size of the spreadsheet
- Degree of understanding and documentation of the spreadsheet requirements by the developer
- Uses of the spreadsheet output
- Frequency and extent of changes and modifications to the spreadsheet
- Development, developer (and training) and testing of the spreadsheet before it is utilised
- Logic errors in which inappropriate formulae are created and generate abnormal results.

The more complex a piece of code the greater the possibility of error. To make end user developers more productive and technically competent requires both training and experience. The training needs to be targeted to meet the required need of each individual end user developer. In order to target correctly, the complexity of the macro code and functions used has to be scrutinised. Software complexity is the investigation of what makes program code difficult for humans to comprehend [114]. There are several well-documented software complexity measures in the literature such as McCabe's cyclomatic number [89], Halstead's programming effort [48], and Oviedo's data flow complexity measures [116]. The motivation of complexity measures is to gauge the correctness, effectiveness and clarity of software and to quantify the costs of testing, maintenance, development time, and number of errors ([83], [114]) .

2.3.2.3.2 Excel Developer Categories

Excel developers can be allocated to five different categories [16]. The allocation is dependent of their knowledge and experience of both excel and the excel macro language Visual Basic for Application (VBA). The categories are Excel User, Excel Power User, VBA Developer, Excel Developer, and Professional Excel Developer.

Every developer is required to make use of certain features in order to be qualified for a certain category. For example, users categorized as *Excel Users* are expected to produce spreadsheets that contain some repetitive calculations and functions like Sum(), Average(), Count(), Max(), Min(), and some other features like storing lists, producing pivot tables and

charts. Users assigned the category *Excel Power Users* should present wider understanding of excel functionality using more advanced functions like Financial functions, Statistical functions, Text functions and create more complex spreadsheets for own use and helps develop and debug colleagues spreadsheets, together with occasional development of VBA macro codes when appropriate. As the category goes higher, developers are required to construct efficient and maintainable applications by making the best use of excel's functionality and make more extensive use of VBA macro code constructs, together with some other programming languages and applications to enhance their excel solutions. Developers with the highest category of Professional Excel Developers are required to have knowledge of all mentioned features and have the capability to use other features like third party ActiveX controls, automate other applications, connect to different databases and use programming languages like C/C++ for fast custom worksheet functions and much more. Table 2.6 adapted from [16] summarizes the categories. Table 6.4 summarizes the categories together with a description and the level of knowledge (Usage) required for every category.

Category	Description
Excel User	<ul style="list-style-type: none"> - Store lists - Simple repetitive calculations - Some worksheet functions - Pivot tables - Charts
Excel Power User	<ul style="list-style-type: none"> - Wide understanding of excel Functions - Create complex spreadsheets for own use and helps develop and debug colleague's spreadsheets. - Occasional use of VBA code from macro recorder or the Internet.
VBA Developer	<ul style="list-style-type: none"> - Extensive use of VBA - Typically they are Power Users who started to learn VBA or Visual Basic developers who switched to VBA development - Often lack sufficient knowledge of Excel to make the best use of its features.
Excel Developer	<ul style="list-style-type: none"> - Constructs efficient and maintainable applications by making the best use of excels' built-in functionality, augmented by VBA when appropriate. - Confident at developing excel based applications for both their colleagues and as part of a development team. - Constrained by their reluctance to use other programming languages and applications to augment their excel solutions.
Professional Excel Developer	<ul style="list-style-type: none"> - Design and develops excel based applications and utilities that are robust, fast, easy to use, maintainable and secure. - Excel forms the core use of their applications, but they include any other applications and languages that are appropriate <p><i>For Example:</i></p> <ul style="list-style-type: none"> - They might use third party ActiveX controls - Automate other applications. - Use Windows API calls. - Use ADO to connect to external Databases. - Use C/C++ for fast custom worksheet functions (DLL/XLL add-ins in XLM macro sheets). - Use VB6 or VB.net for creating object modules and securing their code. - Use XML for sharing data over the Internet.

Table 2.6 Excel developer categories

2.3.2.3.3 Justification of Spreadsheets as a Workbench for Classification

Excel spreadsheets have been used as a workbench for classification in this research project due to a number of factors. Many companies rely on spreadsheets as a key tool in their financial reporting and operational processes. As a result, the use of spreadsheets is an integral part of the information and decision-making framework for these companies [123]. [65] indicate that spreadsheets are still an extremely popular desktop application in workplace today and estimate that this tool is installed on the desktops of four hundred million users. “Whether you work at an accounting firm, a marketing company, an auto dealership, a school attendance office, a manufacturing plant's human resources department, or an office associated with city, county, state or federal government, chances are, you'll be called upon to use and learn Excel” [138].

[15] state that working with Excel actually has two important benefits: (1) a knowledge of excel is important in workplace and the job market, and (2) setting up spreadsheet models and analysing the results also provides useful insights into the implications of financial decisions. Users of excel spreadsheets have become more sophisticated. Once used to support simple functions such as logging, tracking and totalling information, spreadsheets with enhanced formulas and built-in advanced features are now used to support such business functions as complex valuation models [123]. “The use of macros and multiple spreadsheets which are linked together allows users to build very complicated – and sometimes convoluted – models and other business functions with minimal or no documentation” [123].

The advantage of Excel VBA reporting type applications is that the built-in charting engine provides a quick and easy mechanism to high quality presentation [16]. The VBA development environment provides productivity-enhancing features that can be used to link with other Microsoft Office products as well as querying external databases and access to email applications.

Due to these factors, a successful spreadsheet classification project can be of benefit to as wide a range of end users as is possible, improving their development activities via allowing for precise tailor training activities for future spreadsheet development.

2.4 Research Ethics

The study was undertaken taking into account the modern workplace and workplace surveillance. Ethical issues encountered in the monitoring and data collection process and reporting the study results are considered. Institutions and business organizations generally use workplace surveillance as a way of monitoring the activities of their employees [3]. Workplace monitoring has been part of the work environment for a number of years and is likely to continue to proliferate with increased sophistication in software surveillance tools [107] organizations might justify the use of surveillance in the workplace for various reasons. According to survey conducted by the American Management Association (2000) the top most reported reasons for developing surveillance include: acquiring information for performance reviews, guaranteeing legal compliance, and controlling costs. Other reasons include protection of business information, security, and safety [27, 59].

Ethics have climbed to the top of the business agenda because the risks associated with inappropriate behaviour have increase in their probability and in their potential negative impact on the organization [128]. “Through advanced computer technology, employers can continuously monitor employees’ actions without the employee even knowing he or she is being watched. The computer’s eye is unblinking and ever present. Sophisticated software allows every minute of the day to be recorded and evaluated” [107].

The participants of this research project, who are students of the University of Glamorgan were fully informed in advance of the type of data will be collected, how it will be collected and processed, and what are the expected results of the study. There are a number of ethical principles considered in this study:

1- Informed Consent

Informed consent is used throughout the course of this research. Potential participants were clearly informed of any features of the research that might influence their willingness to take part in the study. They were provided with sufficient information to enable them to make informed decision of their participation. An oral consent in the present of a witness was provided that was to adequately brief the individuals and give them the chance to ask question about the study and the nature of data collected. Table 2.7 adapted from [128] demonstrates this in practice and presents all the features that were discussed with the participants.

Checklist of Requirements of Informed Consent	
About the nature of research	
What is the purpose of the study?	To produce a monitoring system and utilize it to provide an automatic classification for end users developing applications in the spreadsheet domain.
Who is or will be undertaking it?	The research will be undertaken by Mohammed Mhereeg from the Computing and Mathematical Science Department as part of a PhD research project.
Who is being asked to participate?	Student from the University of Glamorgan/ Computing and Mathematical Science Department doing an Excel application course as part of their studies.
How far has the research project progressed?	The research project is in its data gathering stage. This stage should continue for 6 weeks before moving to the analysis stage.
About the requirement of taking part	
What type of data will be required for those that agree to take part?	Data will consist of the contents of excel spreadsheet created or updated by the users. The spreadsheets will be monitored and data will be collected in terms of file properties including the Author's name of the spreadsheet, and in terms of the number and type of functions and formulas used. Excel VBA macros will also be monitored to collect data on the number and type of program constructs present in the code.
How will these data items be collected?	The data will be collected via Windows Service based agents that will execute automatically and run continuously without requiring any user intervention.
How much time will be required and on how many occasions?	The agents will be running in the backgrounds collecting the data as long as the computers are being used by the participants. There is no time requirement on the participants to take part in the study.
What are the target dates?	The target dates required for participation will be from 15 th October 2009 to November 2009.
About the implications of taking part and participants' rights?	The privacy of participants is respected and personal information is kept confidential and secure. Participation is voluntarily based and participants may withdraw from the research at any time during the data collection process. The data provided will not be used for any other purposes than those explained to participants.

Recognition that participants have the right to decline to answer any questions or set of questions or to be observed in particular circumstances.	Participants are informed in the beginning about their right to decline participation or to withdraw consent at any point in time with no consequences to their future care or treatment.
Recognition that participants may withdraw at any time.	Participants are given partial control over the monitoring system, they can switch it off completely at any point of time, or temporarily if they do not like to be monitored in some particular circumstances.
What are the consequences of participating – possible risks and expected benefits?	Accurate classification of spreadsheet developers and allowing for precise tailor training activities for future spreadsheet development. There are no risks associated with participating in the research project.
About the use of data collected and the way in which it will be reported	
Who will have access to data collected?	The research author and supervisory team
How will the results of the research be disseminated?	The data will be used to categorize end user developers according to their knowledge of both excel, and excel macro language Visual Basic or Applications (VBA). The classification is based on a criteria explained in tables 2.3 and 6.2. The data collected will not be used for any other purposes than what prescribed to the participants. The results can be used by management to improve excel spreadsheet developers productivity.
How will assurance about anonymity and confidentiality be observed at this stage?	Data will be stored in an Oracle database that will be secure and accessible only to the author of the research and the supervisory team.
What will happen to the data collected after the project is completed?	The data will be kept for a maximum of two years after the project is completed. The data will then be wiped out from the database.
Where data to be preserved, what safeguards will be 'built in' to safeguard the future anonymity and confidentiality of participants?	Data will be kept in a secure Oracle database.

Table 2.7 Checklist of Requirements for Informed Consent

2- Openness and honesty

The discussions held with the consent participants were open and honest about the research. Participants were informed with the specific purpose of the study at the beginning avoiding any kind of deception.

3- Right to withdraw

Participants were clearly informed that they have the right to withdraw at any time without penalty.

4- Protection from harm

Despite there is no physical or psychological harm results from the research procedures, this consideration was an over-riding factor in the relevant stages of the project, such as the monitoring and data collection procedures, and the dissemination and use of findings. The possible consequences of these stages were assessed and every measure taken to ensure safety and prevent adverse effects on participants.

5- Debriefing

Potential participants were provided at the beginning of the study with sufficient information of its purpose as well as its procedures and possible results. The information was presented verbally. That was to give the participants opportunities to ask questions about the study to help them decide if they want to take part in the research.

6- Confidentiality

The anonymity and privacy of participants were respected and personal data was kept confidential and secure throughout the data collection and results reporting phases of the study. The identities of participants were *anonymised such that it was not possible for the University, its staff, students, or researchers to identify an individual from the data and any other data held in the database*

Chapter 3 - The MultiAgent System Architecture of MACS

3.1 Introduction

This chapter presents the agent-based architecture of MACS Classification System. It defines the positions and roles of each agent of the system, and shows how the .NET Windows Services based agents can offer a hosting environment to the .NET WCF Services to operate in. finally, it discusses how the agents communicate over the network to achieve their goals.

3.2 MultiAgent System Based on SOA

The introduction of web services allowed the developers to create applications that can communicate across platforms and programming languages over a network. Web services use a set of open standard protocols based on XML to expose functions (or methods) on the web as web services and allow clients to discover and make the synchronous calls to the exposed methods.

The Windows Communication Foundation (WCF) that is part of .NET Framework 3.0 introduces an extension to web service technology [119, 151]. The main advantages of this platform are that the WCF services and clients can participate in asynchronous service calls that enables the application to continue operating while the method call runs and, the ability to communicate over a variety of supported protocols, including HTTP, TCP, named pipes, and MSMQ. .NET 3.0 and newer versions offer a great support for applications built in Service Oriented Architecture (SOA) [51, 75]. The Service Oriented Architecture paradigm is presented in Figure 3.1.

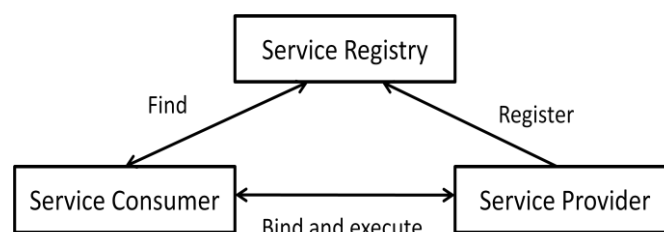


Figure 3.1 Service Oriented Architecture Schema

In the SOA a system consists of the following three components:

Service Provider: is an application that exposes a service, accepts and executes requests from consumers.

Service Consumer: is an application that accesses a service over a transport protocol and, executes its functions.

Service Registry: is a directory that keeps information about exposed services, such as services' addresses, names, parameters, and returned values.

Comparing this paradigm with the MultiAgent architecture proposed by FIPA (*described in 2.2.1.8.2*), one can find that SOA's *Service Registry* plays the same role as FIPA's *Directory Facilitator*, and the *Service provider* and *Service Consumer* are both the agents. These agents according to FIPA are the programs that expose and consume some services. The role of Message Transport System is assigned to the web services or its extensions (WCF Services). The MultiAgent Classification System (MACS) has been built in .NET Framework based on SOA. This gives the system more elastic and moreover the interaction between the agents is simpler and its safety is on the upper level [51, 75].

Figure 3.2 presents the skeleton of the proposed architecture of the MACS multi-agent system based on SOA. The *User Agent* uses the *Service Registry* to manually register and unregister the services of MACS during the design phase of the system. On launch, the *User Agent* queries the *Service Facilitator* requesting the services available in the system for consumption, the agent then supplies the *Database Updater Agent* with the names and addresses of these services. The *Database Updater Agent* starts to communicate with the *Monitoring Agents* using the supplied addresses of the WCF services they expose that is to retrieve the gathered data. Data is then processed by the *Database Updater Agent* and uploaded to the data mining tools for classification purposes.

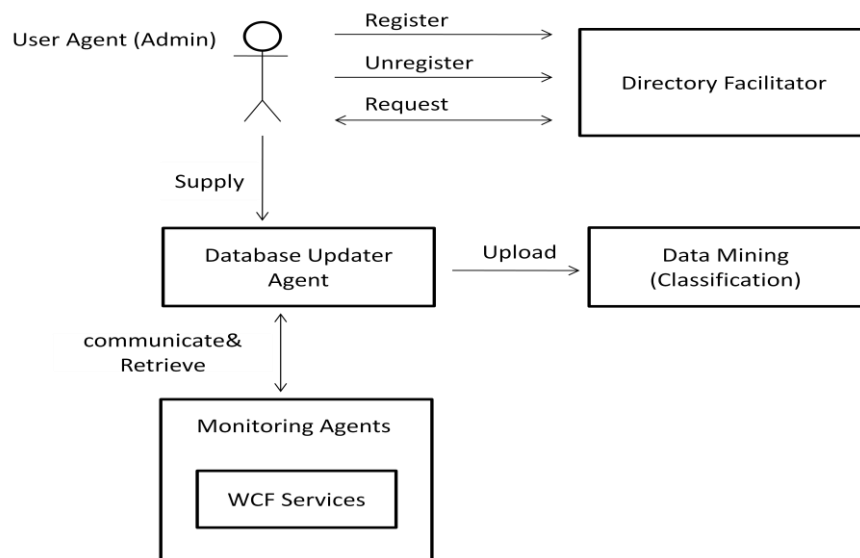


Figure 3.2 Architecture skeleton of the MACS multi-agent based system.

3.3 MACS Multi Agent System

Using the techniques mentioned in the previous section, the Multi Agent Classification System was developed. The aim of MACS is to support the automatic monitoring and categorization of Excel developers' personnel over a network, and allow for precise tailor training activities for future spreadsheet development.

The MACS architecture is composed of the *client* (Master Computer) and a collection of *web-servers*. Each of them having the tools and agents required to undertake particular roles in the system. The web-servers consist of the Monitoring Agents responsible for the data gathering process, and the local data stores needed to save the gathered data. The client consists of a group of agents responsible for establishing the connections with the services offered by the agents hosted in the web-servers, and for the data retrieval, filtering, and transfer to the global database, in addition to the data mining tools responsible for the automatic classification phase of the project.

However, alternatively, one could argue a case of splitting the Monitoring Agent into two separate agents on each web server, one monitoring and saving data into a local data store, and one for waiting for calls from the Database Updater Agent and transferring the contents of the data stores to the client machine for further processing. This could be an alternative design, but this requires introducing an unnecessary extra agent. Thus, the primary motivations for choosing not to perform this split are:

- .NET allows communicating with the Monitoring Agents' exposed services and collecting the gathered data held in the data stores even if these agents are busy running other monitoring activities. In other words, the monitoring and receipt of the incoming calls for data retrieval operations can be managed parallelly.
- Having the WCF services hosted inside the Monitoring Agents allows the creation of long running services that has the capability to survive in the web servers as long as the agents hosting them are available (i.e. running in the background). (*Covered in detail in section 3.3.1*).

Figure 3.3 represents the MACS multi-agent based architecture and the interaction between the agents.

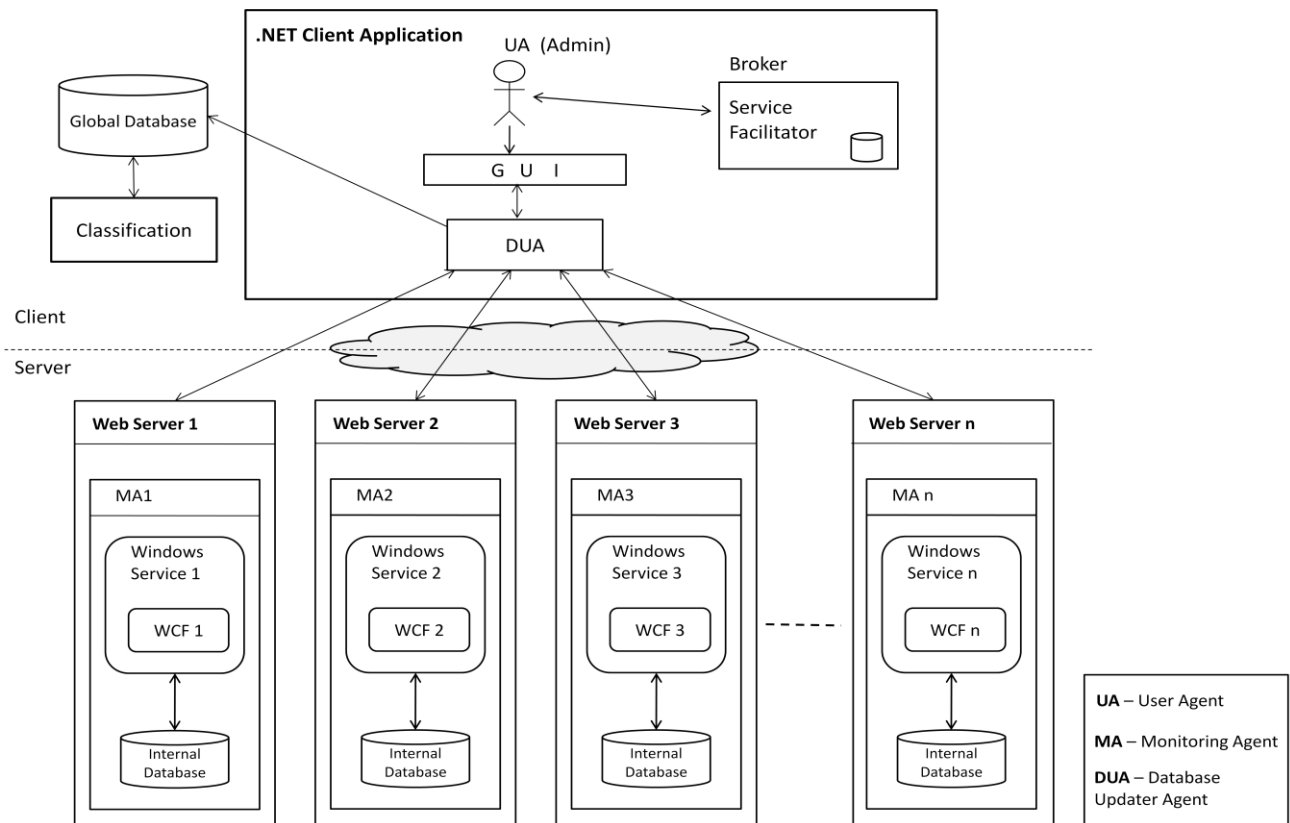


Figure 3.3 MACS Architecture

- The client side application has two types of agents:

User Agent (UA) is the real user (administrator of the system) that assists the communication between the agents. He/she communicates with the Service Facilitator and identifies the available services that the DUA agent can communicate with and consume.

Database Updater Agent (DUA) is the Service Consumer used to communicate with the Monitoring Agents in the server-side application, collects and filters the gathered data, and then stores the data in the appropriate tables within the Global Database (Oracle Server Database).

- The agents on the server side are:

Monitoring Agents (MAs) are the service providers used to achieve the monitoring tasks over the network and supply the DUA agent with the gathered development activities. A .NET Windows Service based agent is installed in every computer to achieve the monitoring, and a WCF service is hosted in every Windows Service process to support the distribution and simplify the access between the client and server agents.

The Service Facilitator is the component that assists the communication between the agents. It keeps the information about the services exposed by the MA agents in the system, like the services' names, addresses and, the type of returned values, and respond to the User Agent queries. The registration and unregistration of the available services are managed manually by the User Agent.

As MACS design is based on SOA, the use of the Service Facilitator is mandatory for the system in order to provide the discovery process of services to the service consumer. In FIPA architecture the implementation of this component is optional [36]. However, it is very important in MACS because it is the central store that holds all the details of the MAs exposed services, these details are required by the DUA to manage the communication and retrieval of data.

WCF technology is used to manage the communication between the agents because of the built-in support for the multi-protocol request-response communication, and because the WCF/SOAP reliable messaging provides end-to-end message transfer reliability between SOAP endpoint of the services [156]. Due to these advantages, the WCF technology plays the role of *the Message Transport System (MTS)*.

The Agent Management System (AMS) is mandatory in FIPA specification [36]. It is used to store the agent details like the agent names and addresses in order to be used for the communication process [36, 37]. In MACS this functionality is omitted, because the agents do not take part in the communication process, this task is achieved through the addresses of the services these agents expose regardless of the locations (computers) in which the agents are installed over the network. This shows that this component is unnecessary in MACS system.

Data collected from the various resources over the network and stored in Oracle tables is used at the end of the monitoring process by Oracle Data Mining tool to provide the classification of the Excel spreadsheet developers. (Covered in detail in chapter six titled the Evaluation and Results).

3.3.1 .NET Widows Services based agents as WCF Service Hosts

.NET Windows Service based agents (MAs) have been used as a hosting environment for the WCF services. This comes with several benefits, for instance this allows the creation of a long-running services that run in the background as long as the MAs are running. In this case,

the life time of the WCF service is controlled by the Windows Service Control Manager (WSCM) and tied up to the life time of the MAs hosting them; this allows automatic starting of the services, so that as soon as the Windows Operating System starts, the services will be started. Recovery is also provided by the WSCM because of its built-in support feature that restarts services when failures occur, thus this ensures the *continuous* operating and availability of the WCF services as long as the MAs are alive. This in turn will add the Proactiveness property to the MAs and make their belonging data stores accessible over the network by the DUA any time during the monitoring process.

3.3.2 Agent Communications in MACS

The communication between the agents in MACS system is based on XML SOAP messages. The server-side agents containing the WCF services are hosted within a web server and TCP is used as the transfer protocol between the client and server applications. At first, the User Agent registers all the WCF services in the Service Facilitator database, the SF is then used as the broker component between the UA, DUA, and the MAs agents. During the communication process, the UA queries the SF database about the services available in the server-side of the system, the SF provides the UA with a list of the names and addresses of the services. The UA then supplies the DUA with these details. As the DUA knows this information, it communicates asynchronously with the MA agents through the addresses of these services for the data retrieval process. Data is received in a string format, filtered and then stored in the appropriate tables on Oracle server database.

Chapter 4 - Design of the .NET Software Agents of MACS

Prometheus methodology and its accompanying Prometheus Design Tool (PDT) have been used for specifying and designing the agents needed to construct the Multi-Agent System (MACS). This chapter demonstrates how Prometheus through its three phases (Figure 2.3), system specification, architectural design and detailed design can be used to achieve this goal.

4.1 System Specification

This sub-chapter discusses the artifacts and processes required in the system specification phase, the initial phase of the Prometheus methodology.

4.1.1 Goal Specification

4.1.1.1 Brief Statement of the Problem Scenario

The system will automatically and continuously monitor end-users developing application in the spreadsheet domain over the network of the University of Glamorgan and gather their development activities. The data gathered from multiple resources will be collected in an Oracle Server Database for further processing. The processing will lead to the classification of the users based on well-defined classification criteria explained in table 2.6. The MultiAgent Classification System (MACS) will consist of a number of software agents. These agents will monitor the usage of software applications on end-users' machines with any new usage patterns being recorded.

The monitoring agents will be installed manually on the users' computers and when Excel spreadsheets are executed, the agents will listen and gather the contents of these applications.

A typical user may start off as a simple consumer of data and progress through simple charts, formulas, pivot tables, automating other applications, and progress further to use Windows API calls and XML for sharing data over the Internet. The monitoring agent will also detect when a macro/Visual Basic Application (VBA) code has been recorded and report when features are being used, such features in the programme environment include the number and type of: selections, sequences, iterations, procedures, functions, inclusion of ActiveX controls and references to COM object libraries. The agents of MACS will be distributed on the user's

computers over the network and run autonomously in the backgrounds collecting the data as long as these computers are being used by the end users.

4.1.1.2 Identifying Initial Goals

I want to develop a multi-agent based system that automatically monitors the software usage of Excel users' development activities on an individual basis and categorise the usage obtained into different categories using well-defined classification criteria from the literature (Table 2.6). The activity together with the categorization can then be used to recommend training and mentoring in order to make the user more productive with respect to the end-user application development. The system must be able to identify any new application development patterns have occurred and distinguish them from pre-existing patterns; this includes the detection of any Macro/Visual Basic Application (VBA) codes generated by the user. The system must be reliable and gather data from both spreadsheets under development and developed applications and produce the data mining based categorization results either on demand or at regular intervals, i.e. monthly or three monthly.

4.1.1.2.1 Identify System Goals

- Monitor software usage of end-users
- Identify application development
- Identify new usage patterns
- Create users Data Mining mapping
- Identify end-users categories

4.1.1.2.2 Goal Refinement

Goal refinement leads to the question 'how might each goal be achieved'? Answering the question normally requires identifying the sub-goals of the goal under consideration. As the initial goals are refined, similar sub-goals may appear under different parent goals. Grouping these sub-goals provides the basis of what is called 'functionalities' or 'roles' – chunks of behaviour of the system. Below shows the expanded list of the initial goals and associated sub-goals, and the initial arrangement of these into groupings.

- Monitor software usage of end-users
 - Identify users
 - Identify packages used
 - When packages used
 - Obtain usage level
- Identify application development
 - Track creation of programme code
 - Track automatic code generation
 - Track generation of hand coding
- Identify new usage patterns
 - Obtain new object usage
 - Obtain code structures
 - Determine new code structures
 - Determine existing code structures
- Create users data mining analysis
 - Get users development efforts
 - Calculate development plane
- Identify users categories
 - Get skills profile of end users
 - Produce categorization results

After refining of goals takes place; these goals have then been rearranged, moving similar goals together, and adding some extra goals to some particular groupings that are lacking some aspect. As this task is achieved, the original set of goals and sub-goals form a network of connected goals. Goals are represented by ovals and arrows join goals to sub-goals. Figure 4.1 shows the goal overview diagram for the MultiAgent Classification System.

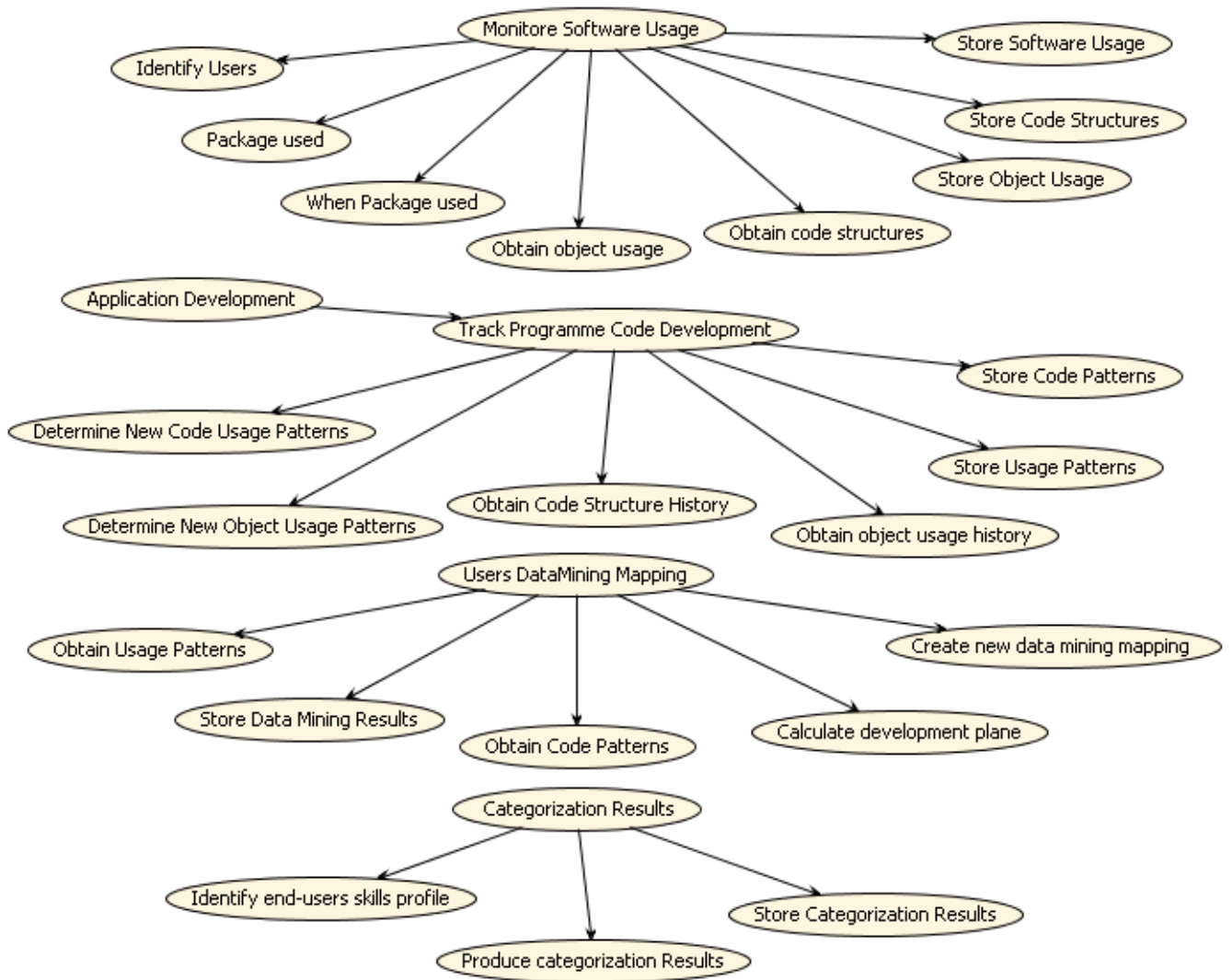


Figure 4.1 Goal Overview Diagram

4.1.2 Roles

Roles can be described as a block of behaviour, which includes a grouping of goals, as well as percepts, actions and data related to the behaviour [117]. The input is represented as a percept and the output as an action. However, grouping of related roles can eventually lead to determining the agent types and their responsibilities. Roles are described by rectangles, goals with ovals, and actions with a rectangle extended with a triangle pointed to the right.

At this stage goals are grouped into cohesive units and assigned to roles which are intended as relatively small and easily specified chunks of agent functionality. The percepts and actions are then also assigned to the roles appropriately to allow the roles to achieve their goals. This is done using the 'System Roles Diagram'.

For example figure 4.2 shows the “*Software Usage Role*” is responsible for the goals to *Identify Users*, *Packages Used* and, *When Packages Used* (i.e. File Properties) of any files used by end users. To achieve this goal the role needs the inputs *User logs on*, *User Creates File*, or *User Updates File* and should perform the action *Write Software Usage to a local store*.

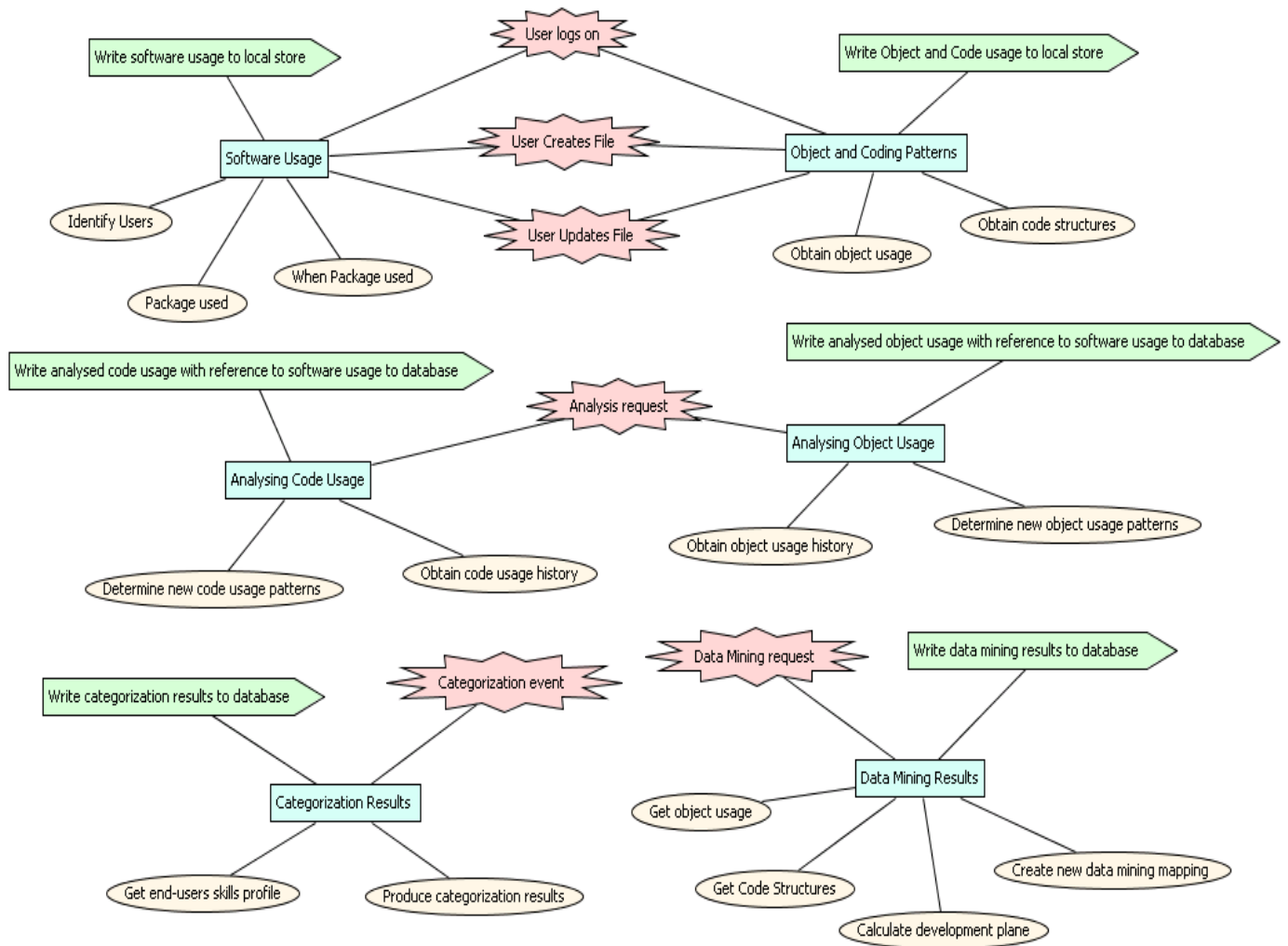


Figure 4.2 System Roles Diagram

Once roles have been identified, role descriptors can be developed. The below shows the role descriptors have been developed on the basis of the goals grouped under each role.

The percepts *User logs on*, *User creates file* and *User updates file* all trigger two roles: “*Software Usage Role*” and “*Object & Coding Patterns Role*”. They carry out the actions: *Write software usage to a local store* and *Write Object and Coding patterns to local store* respectively, and depending on the percept the appropriate goals will be satisfied.

Software Usage Role Descriptor	
Name	Software Usage
Description	The role monitors the software being used on a daily basis by end users
Percepts	User logs on, User creates file, User updates file
Actions	Write software usage to a local store
Data used	Software used
Data produced	Software used local store
Goals	Identify users, Package used, When packages used

Object and Coding Patterns Role Descriptor	
Name	Object and Coding Patterns
Description	The role monitors the Object and Code patterns of a software being used on a daily basis by end users
Percepts	User logs on, User creates file, User updates file
Actions	Write Object and Coding patterns to a local store
Data used	Object and coding patterns used
Data produced	Object and coding patterns used local store
Goals	Obtain object usage, obtain code structures

The percept *Analysis Request* is performed in an ad-hoc manner to trigger the roles “*Analysing Object Usage*” and “*Analysing Code Usage patterns*”. They carry out associated actions of *Writing analysed object usage with reference to software usage to a database* and *Writing analysed code usage with reference to software usage to a database* respectively to meet the roles’ assigned goals.

Analysing Object Usage Role Descriptor	
Name	Analysing Object Usage
Description	For each user build up a history of the object usage and determine new object usage patterns
Percepts	Analysis request (ad-hoc request)
Actions	Write analysed object usage with reference to software usage to a <i>database</i>
Data used	Object usage patterns used
Data produced	Object usage patterns used database
Goals	Obtain object usage history, determine new object usage patterns

Analysing Code Usage Role Descriptor	
Name	Analysing Code Usage
Description	For each user build up a history of the code usage and determine new code usage patterns
Percepts	Analysis request (ad-hoc request)
Actions	Write analysed code usage with reference to software usage to a <i>database</i>
Data used	Code usage patterns used
Data produced	Code usage patterns used database
Goals	Obtain code usage history, determine new code usage patterns

The ad-hoc *Data Mining Request* percept triggers the role “*Data Mining Analysis*”. The action *Write data mining results to database* is to be taken to satisfy the goals assigned to this role.

Data Mining Analysis Role Descriptor	
Name	Data Mining Results
Description	Get object and code usage patterns of users and calculate development plane
Percepts	Data Mining request (Ad-hoc request)
Actions	Write data mining results to a <i>database</i>
Data used	Analysed object and code usage patterns
Data produced	Data mining results database
Goals	Get object usage, Get code usage, calculate development plane, create new data mining mapping

The percept *Categorization Event* triggers the role “*Categorization Results*” that is performed after the “*Data Mining Analysis Role*” has been achieved. The end users skills profile is retrieved and examined. Resulting in, a categorization results are to be produced that shows the users’ categories have been developed over a period of time.

Categorization Results Role Descriptor	
Name	Categorization Results
Description	Obtain skills profile of users for categorization purposes
Percepts	Categorization event (Ad-hoc request)
Actions	Write categorization results to a <i>database</i>
Data used	End-users skills profile database
Data produced	Categorization results database
Goals	Get end-users skills profile, produce categorization results

4.1.3 Interface Description

Agents are often situated in an environment that is dynamic and change rapidly. Thus, agents need to respond to these changes while trying to achieve a goal. Interface description deals with how agents interact with and affect the changing of the environment in which they are situated.

4.1.3.1 Percepts and actions

Percepts can be defined as the expected incoming information that will be available while the agent is running, the agents in this case are required to react to this change in the environment and action is to be taken to fulfil the task under consideration. Two types of incoming information have been considered. The first one happens via some form of event (user logs on a computer), in the second the agent is required to seek the data via monitoring the environment and listening to any files accessed by the users.

The list below illustrates the percepts and actions identified for the MACS system.

Percepts:

- User logs on
- User creates file
- User updates file

Actions:

- Store software usage
- Store object & code usage patterns
- Store data mining results
- Store categorization results

4.1.3.2 Data

As roles are developed, it is also important to consider both the data is to be created or consumed. Data produced and used by MACS system are:

Software Used Database - contains data relating to the software applications being used by each end user, e.g. Author, Date Created, Date Last Saved, Last Saved By,... etc. it also contains information on the object patterns and code structure present in end users code, e.g.

selection statements, iterations, procedures, functions (Table 4.1 shows the software usage and object & code patterns). This database keeps track of the users' development activities.

Software Usage Analysis Database - contains the total analysed usage of the users' object and code patterns of developed applications. This usage will be used for mining the users' development activities. *Data Mining Results Database* - contains records of end users skills profile. *Categorization Results Database* - contains information relating to end users categorization results.

Software Usage	Object & Code Usage Patterns			
File properties	Functions	Iterations	Variables & Constants	Applications & Languages
Author	Database	Do ... Loop Until	Constants	ActiveX Controls Automated applications Windows API calls ADO C/C++ VB6 or VB.net XML
Company	Lookup & Reference	Do ... Until Loop	variables	
Version	Date & Time	Do... Loop While		
Date Last Saved	Math & Trig	Do ... While Loop		
Date Created	Financial	For ... Next		
Comments	Information	For ... Each		
Document	Logical	Selections	Procedures	
Name	Text	If...Then...End If	Function	
Path Name	Cube functions	If...Then...Else...	Sub	
Application	Engineering	End If		
Last Saved By	functions	Select...Case		
Title	Add-ins & automaton			
Subject	functions			

Table 4.1 Software usage and object & code patterns

4.2 Architectural Design

In the System Specification phase, the system scenarios, goals, roles and role descriptors are developed. In the architectural design phase these artifacts will be used for developing the high-level design of the agent system. In Prometheus, all phases of the design interact with each other. Therefore, the architectural design aspects covered in this subchapter interact with each other as well as with the system specifications phase features.

The three aspects that are developed during architectural design are:

1. Deciding on the agent types used in the application.
2. Describing the interaction between agents.
3. Designing the overall system structure using the *System Overview Diagram*.

4.2.1 Deciding on the Agent Types

A major decision is to be made during the architectural design is deciding the agent types the system will contain. Agent types are formed by combining roles, taking into consideration the standard software engineering criteria of coupling and cohesion. The relationships between roles and data stores they interact with are also considered in determining the agent types.

4.2.1.1 Grouping Roles

After roles have been identified, there are many possible solutions in which these roles can be grouped into agents. A good proposed grouping is the one that offers lower degree of dependency or coupling and higher degree of cohesion. [117] Stated two reasons for deciding to group roles into a single agent:

1. The roles seem related – it ‘makes sense’ to group them. For example, the *Software Usage* and *Object & Coding Usage* roles are clearly related.
2. The roles require a lot of the same information. If grouped into a single agent. This can then be represented in internal agent data structure.

They also stated the below reasons for not grouping roles:

1. The roles are clearly unrelated.
2. The roles exist on different hardware platforms.
3. Different numbers of roles are required at run time.

The Data Coupling Diagram is the tool used to develop and assess the proposed groupings. These groupings are then evaluated and refined using the Agent Acquaintance Diagram.

4.2.1.1.1 Data Coupling Diagram

One technique that is used to assess coupling and help in finding groupings which are relatively loosely data coupled is the Data Coupling Diagram. A data coupling diagram consists of all roles formed in the previous phase linked to data that has been identified as necessary for performing the role (not only persistent data, but also data the roles require to fulfil their job).

Rectangles are used to depict roles, and cylinders are to depict data. Direct links are then inserted between roles and data, an arrow from role linked to data indicates the data is produced or written by the role, whereas an arrow from data linked to role indicates that data is used or read by that role. A double-headed arrow indicates that the role both produces and uses the data.

Various design decisions can be made to group roles into agents. A robust rationale for grouping roles is to examine the data sharing, and in particular if roles write to the same database store, it is an indication for grouping them, thus these roles should be in the same agent. Figure 4.3 shows the Data Coupling Diagram with two possible groupings.

The roles *Software Usage* and *Object & Coding Usage Patterns* are both used to populate the *Software Usage Local Data Store*, thus they make up the first agent namely the Monitoring Agent. The roles *Analysing Object Usage* and *Analysing Code Usage Patterns* read data from the *Software Usage Local Data Store*, analyse the data and, then populate the *Software Used Database* and *Software Usage Analysis Database* data stores. These roles make up the second agent named Database Updater Agent.

The *Data Mining Analysis* and *Categorization Results* Roles are not to be grouped. These roles will be performed manually and separately by the User Agent (i.e. Administrator of the system) in an Ad-hoc manner.

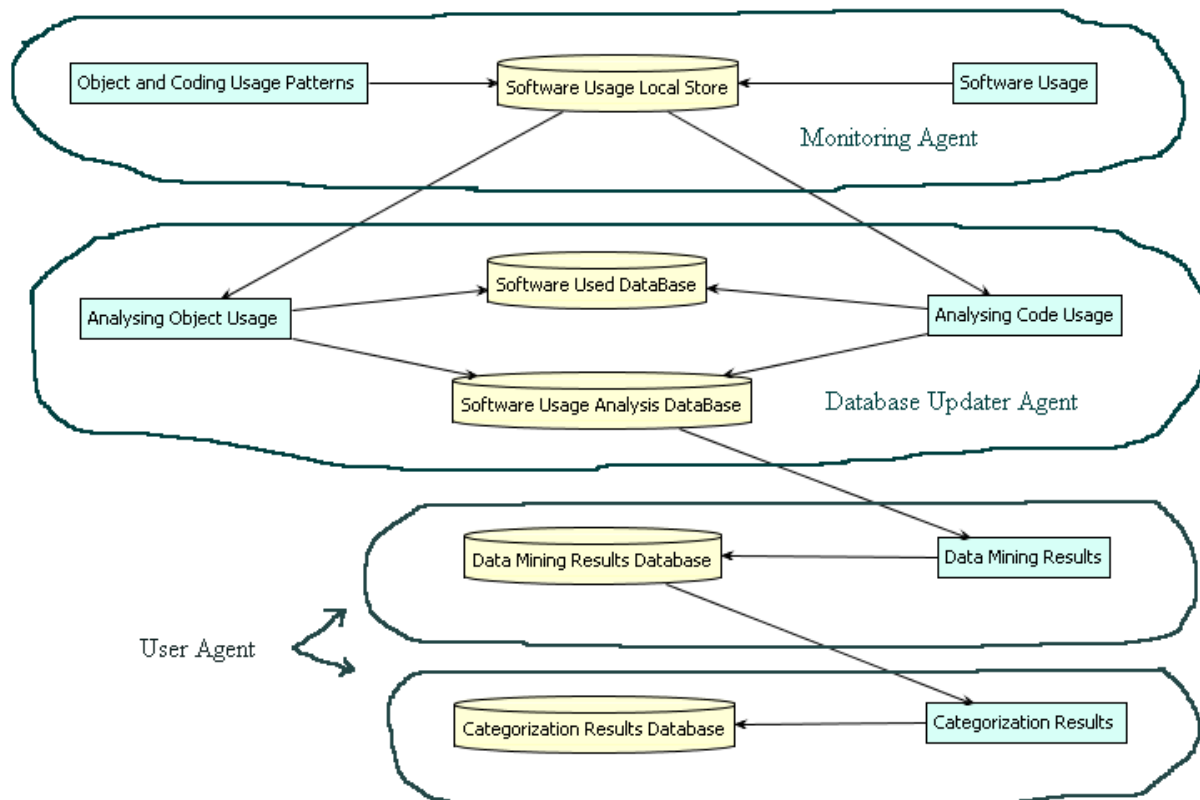


Figure 4.3 Data Coupling Diagram

4.2.1.1.2 Agent-Role Grouping Diagram

In this diagram the roles are grouped into agents. The Agent-Role Grouping Diagram shows the group of roles that come under each agent. Figure 4.4 illustrates that the roles of *Software Usage* and *Object & Coding Usage* as being part of the Monitoring Agent, whereas the roles of *Analysing Object Usage* and *Analysing Code Usage* as being part of the Database Updater Agent.



Figure 4.4 Agent-Role Grouping Diagram

4.2.1.1.3 Agent Acquaintance Diagram

The Agent Acquaintance Diagram evaluates the groupings for coupling and places a link where there is a communication between the agents. These links are automatically created when a message (call) is sent from one agent to another. Figure 4.5 shows that a message is sent from the Database Updater Agent to the Monitoring Agent requesting the collected data the local store contains.

One of the design aims is to produce a system that is as loosely coupled as possible. The Figure below indicates that the design has a low level of coupling with one linkage between the agents and therefore preferable.

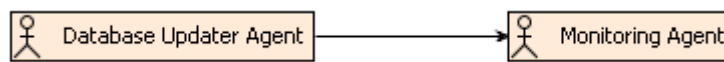


Figure 4.5 Agent Acquaintance Diagram

4.2.2 Developing Agent Descriptors

Once the agents of the system have been decided, Agent descriptors are to be produced. This would provide higher level information contains answers to the following questions [117].

- How many agents of each type will there be?
- What is the lifetime of the agent?
- Agent initialization – what needs to be done?
- Agent demise – what clean up needs to be done?
- What percepts will this agent react to?
- What actions will it take?
- What are the goals of the agent?
- What data does this agent uses or produces?

In addition, each agent should have an extra information includes the name of the agent, a natural language description of what this agent does within the system and a list of the roles that have constructed the agent. Tables 4.2 and 4.3 show descriptors for the Monitoring Agent and the Database Updater Agent respectively.

Name	Monitoring Agent
Description	Monitor software packages used by an end user
Cardinality minimum	One per networked computer
Cardinality maximum	One per networked computer
Lifetime	Instantiated when user logs on. Demise when user logs out Can be started, paused, stopped and restarted manually by the user
Initialization	Listen for packages that are being launched, data collection
Demise	Close open communications
Percepts	User logs on, user creates file, user updates file
Actions	Write software usage to local store, write object & Coding patterns usage to local store
Uses data	None
Produces data	Software usage local store
Goals	Identify users, packages used, when packages used, obtain object usage, obtain code usage
Roles	Software usage, object & code usage patterns

Table 4.2 The Monitoring Agent Descriptor

Name	Database Updater Agent
Description	Collects data gathered by the Monitoring Agents, filters the data, and then connects to Oracle server database for data upload.
Cardinality minimum	One per networked master computer
Cardinality maximum	One per networked master computer

Lifetime	Instantiate manually by the user agent. Demise when user logs out
Initialization	Obtains data gathered by the Monitoring Agents
Demise	Closes open database connections
Percepts	Analysis request
Actions	Write analysed data with reference to software usage to Oracle Database
Uses data	Software usage local store
Produces data	Software used database, software usage analysis database
Goals	Obtain object usage history, determine new object usage patterns, Obtain code usage history, determine new code usage patterns
Roles	Analysing object usage, analysing code usage

Table 4.3 The Database Updater Agent Descriptor

4.2.3 System Overview

Having identified the agents the system will include. The top level structure of the system can be produced using the System Overview Diagram (Figure 4.6). This diagram brings together the agents, percepts, actions, and data stores and shows how they will interact to achieve the system goals. It shows which percepts and actions are associated with which agent. It shows also which data stores are used, which are produced and, which are both used and produced by which agent. Messages that can be sent between agents can also be shown in this diagram. It is arguably the single most important artifact of the entire design process, although of course it cannot be really understood fully in isolation [117]. The System Overview Diagram assists to obtain a general understanding of how the system will function, including the interactions between agents.

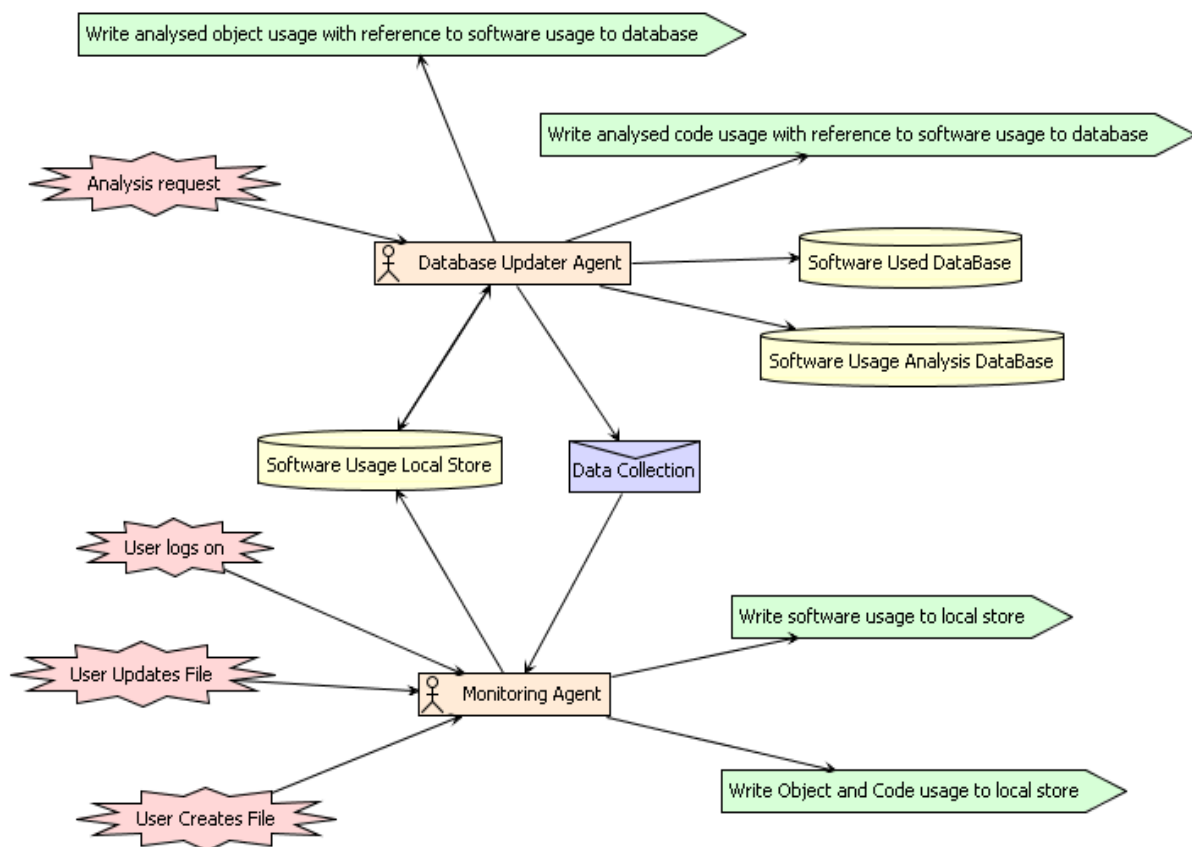


Figure 4.6 System Overview Diagram

4.3 Detailed Design

The detailed design stage deals with the internal behaviour of each agent individually. It takes the artifacts have been developed in the architectural design phase and for every agent develops the capabilities necessary to achieve its goals. The roles of the agents that have been defined in the specification phase can be used to suggest an initial set of capabilities, which can then be refined as desired. The detailed design describes agents in terms of capabilities and the interactions between them. These capabilities are of necessity to the implementation platform more than the preceding steps. The detailed design phase consists of two Agent Overview Diagrams, one for each agent as shown below.

4.3.1 Agent Overview Diagram

The Agent Overview Diagram provides a top level view of the internal structure and behaviour of the agents of the system. In this stage, an Agent Overview Diagram is produced for every single agent appeared in the System Overview Diagram. The Agent Overview

Diagram is very similar to the System Overview Diagram, but instead of showing the interactions between agents within the system, it shows the interactions between the capabilities within an agent. This diagram brings together the capabilities, and all the percepts, actions and data stores linked to an agent within the system. The Agent Overview Diagram also includes the messages required between capabilities in order to realise the behaviour of an agent. Figures 4.7 and 4.8 illustrate graphically the detailed design of the Monitoring Agent and the Database Updater Agent respectively. By looking at these diagrams, it is possible to gain a high level view of how the capabilities of the system will interact to achieve the overall tasks of the agents.

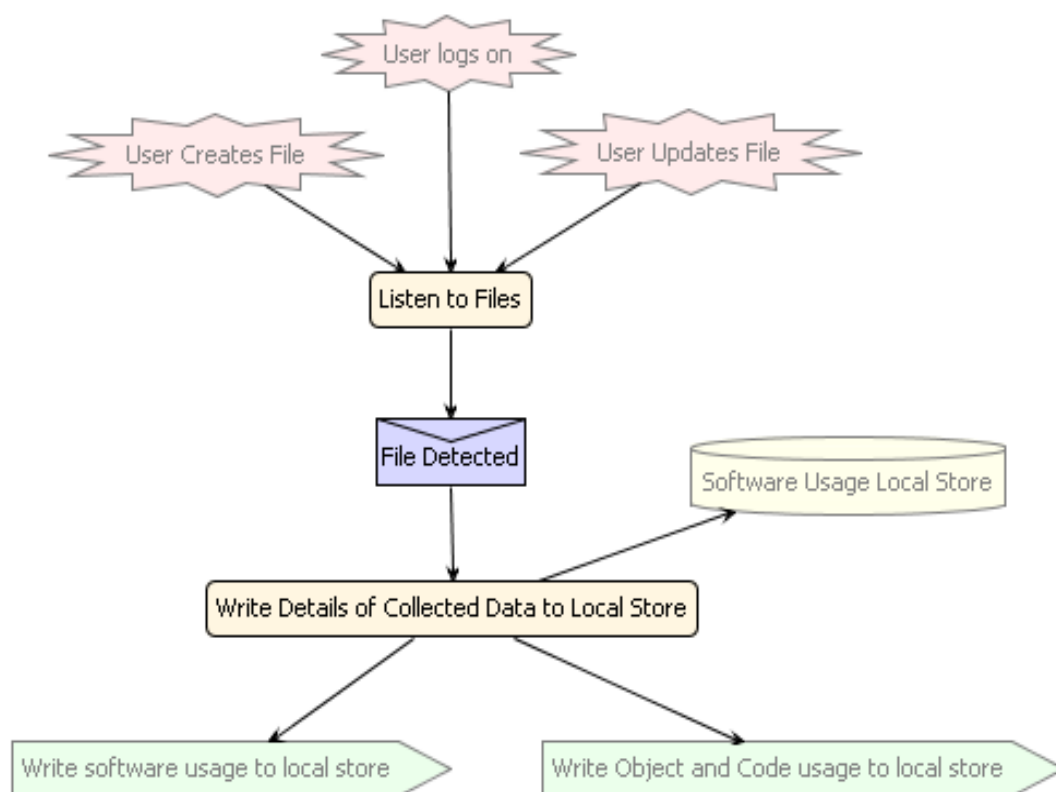


Figure 4.7 Agent Overview Diagram for the Monitoring Agent – Detailed Design

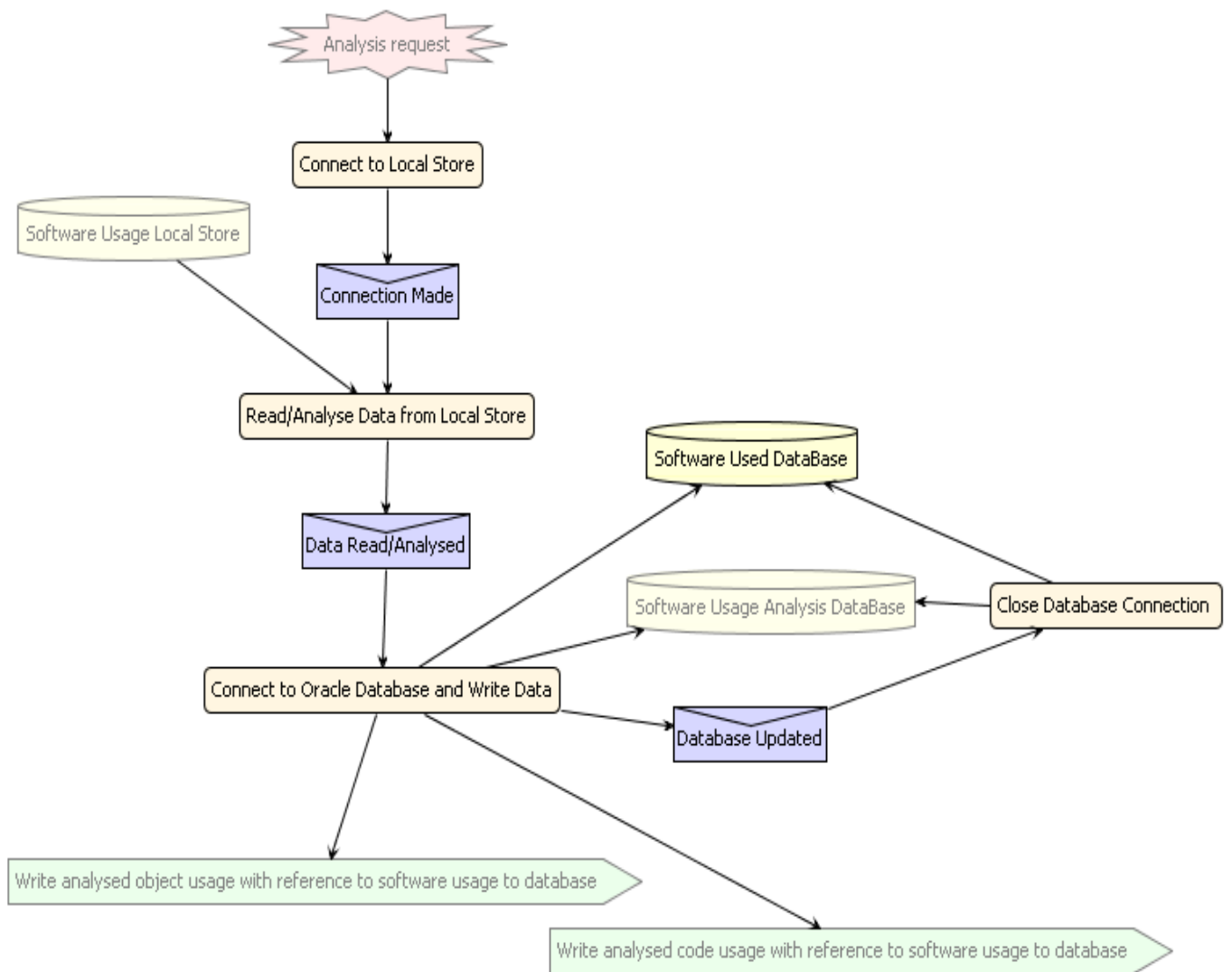


Figure 4.8 Agent Overview Diagram for the Database Updater Agent – Detailed Design

Chapter 5 – Implementation and Testing

5.1 Introduction

This chapter presents a description of MACS prototype. However, MACS constitute of over 15.000 lines of code, thus a detailed description cannot be provided, though every agent involved in the construction of MACS is presented. However, this chapter is divided to five main subchapters. The first subchapter presents the software applications used for the development of the prototype. The second subchapter covers the server side application. The client side application is covered in the third subchapter, some implementation considerations are discussed in the fourth subchapter. The fifth subchapter covers the testing of MACS.

5.2 Implementation of MACS Prototype

The main software applications have been used for the implementation of the prototype are Microsoft's .NET Framework, Visual Studio.NET 2008, and Oracle Server Database. .NET framework was chosen as a development environment within which the agents can be created and communicate to achieve their goals. It provides a feature-rich application that consists of two parts: the Common Language Runtime (CLR) and a unified set of class libraries. These two components provide the execution engine for building .NET applications. The CLR is significant in making the .NET framework, platform and language independent. It also provides other important services such as security, memory management, and exception handling. The class library includes many attractive features, including Consol Applications, Graphical User Interface (GUI) applications (Windows Forms) for smart client applications, Windows Services, and Windows Communication Foundation services which are essentials for developing MACS multi-agent system.

Windows services were chosen as they enable developers to produce long-running executable applications that run in their own windows sessions. This concept was used in the construction of the Monitoring Agents (MAs) that reside in the server-side application to monitor and collect data from end user excel spreadsheets.

Windows Communication Foundation services technology that ships as part of the .NET framework 3.0 and newer versions has been used for the distribution of the agents over the University of Glamorgan's network. This technology is also used to manage the communication between the agents because of the built-in support for the multi-protocol

request-response communication, and because of the provision of WCF/SOAP reliable messaging transfer between the agents. Additionally, the .NET framework enables the asynchronous service calls participation between the WCF clients and services. A high-level performance comparison report [47] between WCF and the other Microsoft distributed communication technologies like: ASP.NET Web Services, Web Services Enhancements (WSE), .NET Enterprise Services, and .NET Remoting shows that in most cases, the performance is significantly better for WCF over the other technologies.

Visual C#.NET that is provided by Visual Studio.NET was chosen as the programming language for the development of the prototype because it is an object-oriented language, and was created specifically for .NET platform. Thus it has the best support for .NET applications. Oracle 10.2g was used as a storage device to store the gathered and analysed data over the life-cycle of the system. The .NET Data Provider for Oracle was the component used to provide access to Oracle Database. Oracle Data Mining is the technology used for the classification of the end users.

The prototype may be divided into the server-side and the client-side software applications as illustrated in *Chapter 3/Figure 3.3*. The sever-side consists of a number of Monitoring Agents running in different computers with appropriate local data repositories. The client-side application consists of the Database Updater Agent, the Service Facilitator to simplify the access between the agents in both sides of the prototype, and Oracle Server Database.

5.2.1 The Server-Side

The Monitoring Agents are installed manually on a number of computers on the network of the University of Glamorgan, one MA per computer. However, the number of MAs (Service Providers) that can be installed is unlimited. These agents are identical and achieve exactly the same task of monitoring and data collection in different machines. Every agent has its own local data store within which the gathered data is stored. The agents are configured to start automatically when the computer within which they are installed boot. They can also be paused, stopped, and restarted manually using Visual Studio .NET Server Explorer. Thus this gives the user control over the agents and the capability to choose if they do not want to be monitored.

WCF services are hosted within the MAs for service exposure and to manage the communication with the Database Updater Agent residing in the client-side. The

communication is based on XML SOAP messages, and TCP is used as the transfer protocol between the client-side and server-side agents. All the WCF services exposed by the MAs are registered manually in the Service Facilitator in order to simplify the access and ensure the availability of services. The following subchapter describes in details the implementation of the Monitoring Agent.

5.2.1.1 The Monitoring Agent

The MA is configured to start automatically when the user logs on and exist as an autonomous windows service process. Because it is a windows service based agent, it will execute continuously and run in the background as long as the computer is switched on. Using the FileSystemWatcher class supplied by .NET (covered in more detail in the section titled '*Detecting File Changes*'), the MA will detect and monitor any excel spreadsheets can be accessed by the user during its lifetime, and store the collected data in a local data store, figure 5.1 shows the basic algorithm of the MA.

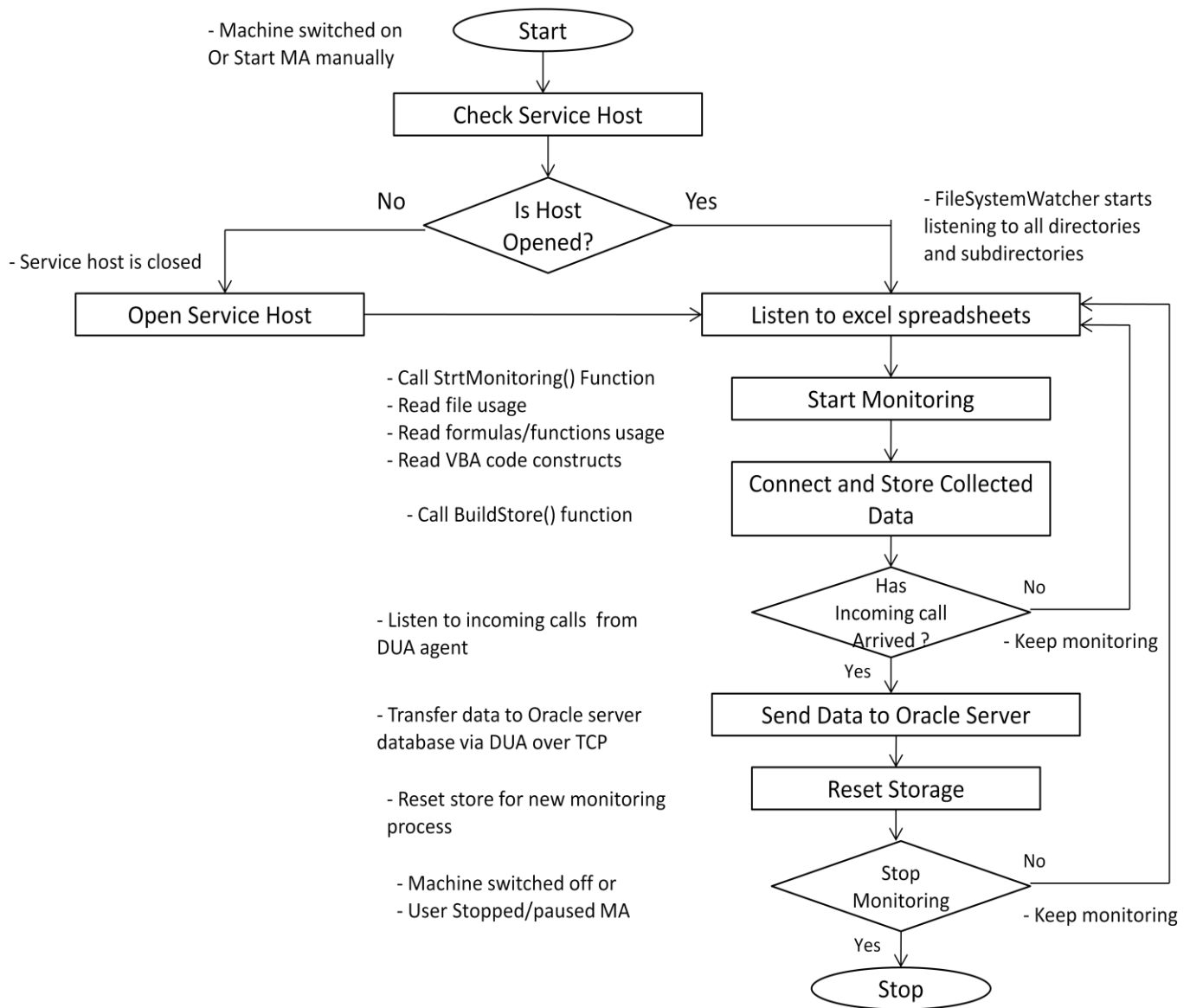


Figure 5.1 The Basic Algorithm of the MA.

The WCF technology is used to provide the communication abilities between the MAs and the DUA agent in the client-side application. A WCF service is built and hosted in every MA agent. The WCF service's interface is defined that consists of one exposed method marked by *OperationContract*; code 5.1 shows the service contract and the service class that implements it. After the interface and its method are defined, they are encapsulated in a class that implements the interface. The method returns a value of type string and responsible for the data return to the client-side application. The method has another task of clearing the *ArrayList* data store, thus the store can be reused for the monitoring activities to come.

```

// step 1: Define a service contract.
[ServiceContract(Namespace = "http://AutoService_StreamTCP_MiniNet.ServiceModel.Agents")]
public interface IAutomation_StreamTCP_MiniNet
{
    [OperationContract]
    string[] RetRangeArrayList();
}

// Step 2: Create service class that implements the service contract.
public class AutomaticService_StreamTCP_MiniNet : IAutomation_StreamTCP_MiniNet
{
    public string[] RetRangeArrayList()
    {
        String[] myArr = (String[])returnArrayList().ToArray(typeof(string));
        ClearArrayList();
        return myArr;
    }
}

```

Code 5.1 WCF Service Contract and the Service Class that implements it

A ServiceHost is constructed that is responsible for opening a communication channel for accepting the incoming calls, provided with an address where the service is located along with the service contract supported by the address, and provided with the supported communication protocol, which is the TCP protocol in this case. When the ServiceHost opens the channel for the service, this allows the DUA agent to make the asynchronous calls to invoke the method provided by the service (Covered in more detail in the section titled ‘*The DUA Agent*’). Thus the contents of the store can be transferred to the client-side application for further processing.

On launch of the MA, the *OnStart ()* procedure executes to check the availability of the service host that is responsible for accepting the incoming calls. If the host is closed, it is opened automatically by the MA. A communication exception is thrown if any communication problems occur (code 5.2 shows checking a WCF ServiceHost).

```

public static ServiceHost svcHost = null;
protected override void OnStart(string[] args)
{
    if (svcHost != null)
    {
        svcHost.Close();
    }
    try
    {
        Uri baseAddress = new Uri("http://J345-MM.uni.glam.ac.uk:9000/ServiceModelAgents/Service");
        svcHost = new ServiceHost(typeof(AutomaticService), baseAddress);
        // Open the service host to accept incoming calls
        svcHost.Open();
        // Start Listening ...
        MA_Listen();
    }
    catch (CommunicationException commProblem)
    {
        Console.WriteLine("There is a communication problem. " + commProblem.Message);
    }
}

```

Code 5.2 Checking a WCF service host

Once this is achieved, the MA starts to listen to any percepts can be risen by users in the environment. When the user launches an excel spreadsheet, the agent automatically detects and opens that spreadsheet (see code 5.3), opens all its workbooks and reads: the file properties (File Usage), all formulas the spreadsheet contains and VBA code constructs (Object and Code Usage), table 4.1 shows the file properties and object and code usage patterns the agent is interested in and required to monitor. More detailed features can be found in appendix C.

This is however the first part of the goal. The MA continuous to monitor the spreadsheet as long as the development process is still running, that is to read any updates the user makes. When the user hits the button *Save* the MA reads: the updated property details, that is in order to record when changes have been made, this achieved via using the property *DateLastSaved*. It also records the entire newly developed Object and Code usage patterns.

```

private Excel.Application ExcelObjf = null;
private void ReadFile(string FileName)
{
    try
    {
        // Construct an Excel Application Object
        ExcelObjf = new Excel.Application();
        // Here is the call to Open a Workbook in Excel
        Excel.Workbook theWorkbook = ExcelObjf.Workbooks.
        Open(FileName.Trim(), 0, true, 5, "", "", true,
        Excel.XlPlatform.xlWindows, "\t", false, false, 0, true, 1, 0);
        // Get the collection of sheets in the workbook
        Excel.Sheets sheets = theWorkbook.Worksheets;
        Excel.Worksheet workSheet = Excel.Worksheet)
        theWorkbook.ActiveSheet;
        foreach (Excel.Worksheet Sheet in sheets)
        {
            for (int i = 1; i <= 300; i++)
            {
                Excel.Range Range = Sheet.get_Range("A" + i.ToString(), "IV" +
                i.ToString());
                System.Array mValues= System.Array)Range.Cells.Formula;
                int ArrayLen = mValues.Length - 1;
                string[] ExcelArray = new string[ArrayLen + 1];
                string[] ExcelArray1 = new string[ArrayLen + 1];
                ExcelArray1 = ConvertToStringArray(mValues);
            }
        }
        TheWorkBook = null;
    }
    catch (NullReferenceException nre)
    {
        Console.WriteLine(nre.ToString());
    }
}

```

Code 5.3 Opening a spreadsheet

Once the monitoring task is complete, the MA connects to the *ArrayList* data store to be used for storing the gathered activity details. However, the store is made available to the DUA incoming calls any time during the process for data transfer, even if the MA is busy running monitoring activities. The WCF is the method by which the DUA agent accesses the *ArrayList* store in the server-side application. Thus it is essentially the interface to the data. Data collected from the *ArrayList* by the DUA is filtered and saved in Oracle server database for further processing (Covered in detail in chapter 5.2.2.1). On completion of the monitoring and data transfer process, the MA reacts to either User logs out or user stops the MA manually. The OnStop() method then executes and closes the ServiceHost as shown in Code 5.4 before the MA stops.


```
protected override void OnStop()
{
    if (svcHost != null)
    {
        svcHost.Close();
        svcHost = null;
    }
}
```

Code 5.4 Stop MA Agents

The functionality of the MA is decomposed into different C# files created all together in a solution explorer that Visual Studio .NET provides. One represents the code needed to listen to and open the excel spreadsheets, and then expose the WCF service for data transfer. One represents the code needed to get the File properties, one represents the code needed to get formulas the spreadsheet contains, one represents the code needed to get VBA Code Constructs, and one represents the code needed to get the regular expressions. An xml configuration file is also created in a separate file to play the role of the communication with the client-side application. This partition enables the easy maintenance of the existing files and the introduction of any new features without affecting the rest of the code. These files are as the following:

- *WCF_HostIn_WS.cs*: contains C# code and represents the main code file for the project. It is used to achieve several tasks: (1) listens to and monitors any excel files being copied manually, opened for updating purposes, or created new in the local path C:\. (2) Executes methods from the other code files during the monitoring process, such as methods that retrieve the file properties, methods that retrieve formulas used, and methods that retrieve code structure patterns. (3) Store the collected data into the ArrayList local store. (4) Opens a communication channel with the UDA agent for the data to be transferred to the client application for further processing.

- *FileProperties.cs*: contains C# code used to read the file properties from the excel file under processing. The file uses the DSOFFile.dll component provided by Microsoft to achieve this task. The complete set of file properties can be found in appendix C, and the complete code for reading file properties can be found in appendix B1.

- *Functions.cs*: contains C# code used to collect the type and number of all functions and formulas the excel spreadsheet under processing may contain. This file has the capability to read data from 249 different excel functions and formulas. The complete set of

formulas/functions can be found in appendix C, and chunks of the code can be found in appendix B2.

CodeConstructs.cs: contains C# code that scans through a VBA code and reads the features the code consists of, such as the Procedures, Functions, For loops, Do loops, and If statements a user has developed. A complete set of the components this file reads can be found in appendix C, and chunks of the code can be found in appendix B3.

RegularExpressions.cs: contains C# code used for matching standards with the code constructs for string parsing and replacement. It scans through VBA code matching texts with patterns, that is in order to find and replace strings that take a defined format. For example, Regular Expressions have been used to parse Public/Private Sub components, Public/Private Function, Do Loops, and For Each components and so on. A complete set of the Regular Expression functions can be found in appendix C, and chunks of the code can be found in appendix B4.

- *App.config*: a configuration file is created to configure where the WCF service that every MA exposes is located, that is in order for the DUA to find it. This configuration allows providing the endpoint of the service and the service behaviour data at the point of deployment instead of the design time. Code 5.5 shows part of the configuration file of a service hosted in a MA running in a machine with a computer name *J345-MM.uni.glam.ac.uk*, the complete code can be found in appendix B5.

```

<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.serviceModel>
    <services>
      <service name="AutoService_StreamTCP_MiniNet.ServiceModel.Agents.AutomaticService_StreamTCP_MiniNet"
        behaviorConfiguration="AgentServiceBehavior">
        <host>
          <baseAddresses>
            <add baseAddress="http://J345-MM.uni.glam.ac.uk:9000/ServiceModelAgents/Service"/>
          </baseAddresses>
        </host>
        <!-- this endpoint is exposed at: net.tcp://J345-MM.uni.glam.ac.uk:8000/ServiceModelAgents/Service -->
        <endpoint address="net.tcp://J345-MM.uni.glam.ac.uk:8000/ServiceModelAgents/Service"
          binding="netTcpBinding"
          bindingConfiguration="StreamedTCPBinding"
          contract="AutoService_StreamTCP_MiniNet.ServiceModel.Agents.IAutomation_StreamTCP_MiniNet" />
        <endpoint address="mex"
          binding="mexHttpBinding"
          contract="IMetadataExchange" />
      </service>
    </services>
    <!--For debugging purposes set the includeExceptionDetailInFaults attribute to true-->
    <behaviors>
      <serviceBehaviors>
        <behavior name="AgentServiceBehavior">
          <serviceMetadata httpGetEnabled="true" />
          <serviceDebug includeExceptionDetailInFaults="False" />
        </behavior>
      </serviceBehaviors>
    </behaviors>
  </system.serviceModel>
</configuration>

```

Code 5.5 app.config Configuration File

The `<system.serviceModel>` section contains the configuration information of the service. The `<services>` section contains the details of the service implemented. The `<service>` element specifies the namespace and class that implement the service.

The `<endpoint>` element provides details of the service that the DUA requires in order to find and communicate with the service. The endpoint consists of three important pieces of information: an address, binding, and contract. The address refers to the location that the MA hosting the service uses to advertise the service. The binding element specifies the mechanism and protocol used to access the WCF web service. A contract defines an agreement of how DUA should communicate with the service. An extra element has been added called *BindingConfiguration* as the streamed technique is used for the data transfer over TCP rather than the default buffered technique over HTTP.

The <behaviours> element indicates that a service metadata is published for the service. The attribute *HttpGetEnabled* is set to “**true**” to allow the metadata to respond to metadata request made by the client using HTTP/GET request. This means that the client application can retrieve the metadata of the service, so that the DUA can use the information provided by the metadata to find/access the service.

5.2.1.2 Detecting File Changes

The FileSystemWatcher is a component provided by Microsoft .NET Framework, which allows developers to connect to directories and watch for specific changes occur to the files and subdirectories within it. This component is utilized by every MA to monitor the local hard drive C:\ and detect any excel spreadsheet changes. However, the MAs have the capability to monitor any hard drives exist in the monitored computer like drives D, E, F...etc. depends of the partition of the machine. The User Agent is responsible for changing the code via replacing the path C:\ (*watcher.Path = "C:\\";*) with any other paths the MAs are required to monitor, for instance, when changing from path C:\ to D:\, the code-line should look like (*watcher.Path = "D:\\";*)

Upon start, the MA instructs the FileSystemWatcher class to start the monitoring process and then the data collection process that is managed by the MA follows whenever a new excel file is created or an existing file is updated. Code 5.6 shows the FileSystemWatcher initialization.

As shown below, a FileSystemWatcher object is created and its properties are set to include all subdirectories of drive C:\. There is a restriction on the file type that it looks only for changes on files with the extension *.xlsx*; this prevents the event handler of being called unnecessarily. The event handler is added to identify the method needs to be called whenever a file is created or updated. Every time the event handler is executed, it calls the method named *Start_Monitoring()* that collects the File Usage attributes, Object and Code Usage Patterns, and some other features like charts and pivot tables and then the local store is populated with the collected data. The *EnableRaisingEvents* property is set to “**true**” to initiate the monitoring process.

```

public void MA_Monitor()
{
    FileSystemWatcher watcher = new FileSystemWatcher();

    watcher.Path = "c:\\";
    watcher.NotifyFilter = NotifyFilters.FileName | NotifyFilters.LastWrite
        | NotifyFilters.DirectoryName;
    // Watch excel 2007 spreadsheets
    watcher.Filter = "*.xlsx";
    watcher.IncludeSubdirectories = true;
    // Add event handlers.
    watcher.Created += new FileSystemEventHandler(OnChanged);
    watcher.Changed += new FileSystemEventHandler(OnChanged);
    // Begin watching.
    watcher.EnableRaisingEvents = true;
}
// Define the event handlers.
private void OnChanged(object source, FileSystemEventArgs e)
{
    // Monitor when a file is created or changed
    Start_Monitoring(source, e);
}

```

Code 5.6 FileSystemWatcher Initialization

5.2.2 The Client-Side

The client-side application initiates the connection with the server-side application and consumes the WCF services that its agents expose. The WCF client is the component used to manage the communication. The UA is responsible for retrieving the metadata manually from the WCF services and use it to create a WCF client that the DUA (i.e. Service Consumer) can use to access the services. These services must be up and running in order for the metadata to be obtained. The UA utilizes the *ServiceModel Metadata Utility* tool provided by .NET Framework to achieve this task. This command-line tool obtains the metadata from the services running in the web server application and generates the required client proxy in C# language. In addition to creating the client proxy, the tool also generates the configuration file that enables the DUA agent to connect to the services using their endpoints this file provides. The following example shows how the tool can be used with the appropriate switches to generate the proxy file together with the configuration file for the service located in the below URI address:

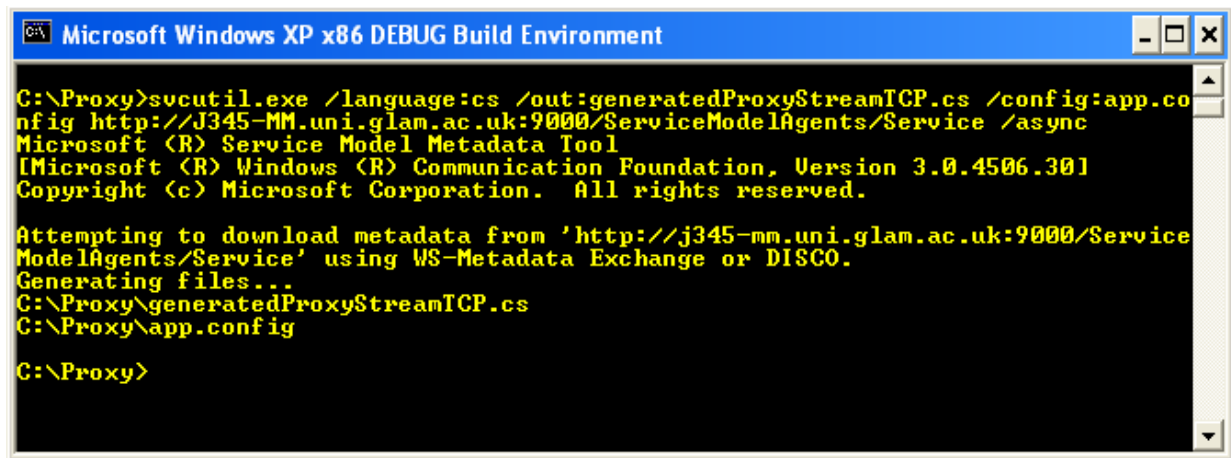
```

svcutil.exe /language:cs /out:generatedProxyStreamTCP.cs /config:app.config http://J345-MM.uni.glam.ac.uk:9000/ServiceModelAgents/Service /async

```

The /async switch generates an asynchronous client version of the service. This can be done for any number of services the client application wants to connect to. A metadata for multiple

services can be obtained at the same time using the same Svcutil.exe command shown above via just adding the addresses of the services following each other. Figure 5.2 shows the outcome of the command.



```
C:\Proxy>svcutil.exe /language:cs /out:generatedProxyStreamTCP.cs /config:app.config http://J345-MM.uni.glam.ac.uk:9000/ServiceModelAgents/Service /async
Microsoft (R) Service Model Metadata Tool
[Microsoft (R) Windows (R) Communication Foundation, Version 3.0.4506.301]
Copyright (c) Microsoft Corporation. All rights reserved.

Attempting to download metadata from 'http://j345-mm.uni.glam.ac.uk:9000/ServiceModelAgents/Service' using WS-Metadata Exchange or DISCO.
Generating files...
C:\Proxy\generatedProxyStreamTCP.cs
C:\Proxy\app.config

C:\Proxy>
```

Figure 5.2 the Svcutil.exe command

The *generatedProxyStreamTCP.cs* C# proxy file and the *app.config* configuration file are both added manually to the client project within the Solution Explorer in Visual studio .NET. These files are the keystone that will be relied on to manage the communication process. Both the proxy and configuration files can be found in appendices D1 and D2 respectively.

A .NET Windows Form based GUI has been constructed. This GUI is used by the UA to query the SF database for the available services the DUA can request data from. The *Call Service* button is used to start the DUA agent to call the selected service in the ComboBox area. Figure 5.3 shows a successful call to the service named J345_mm.

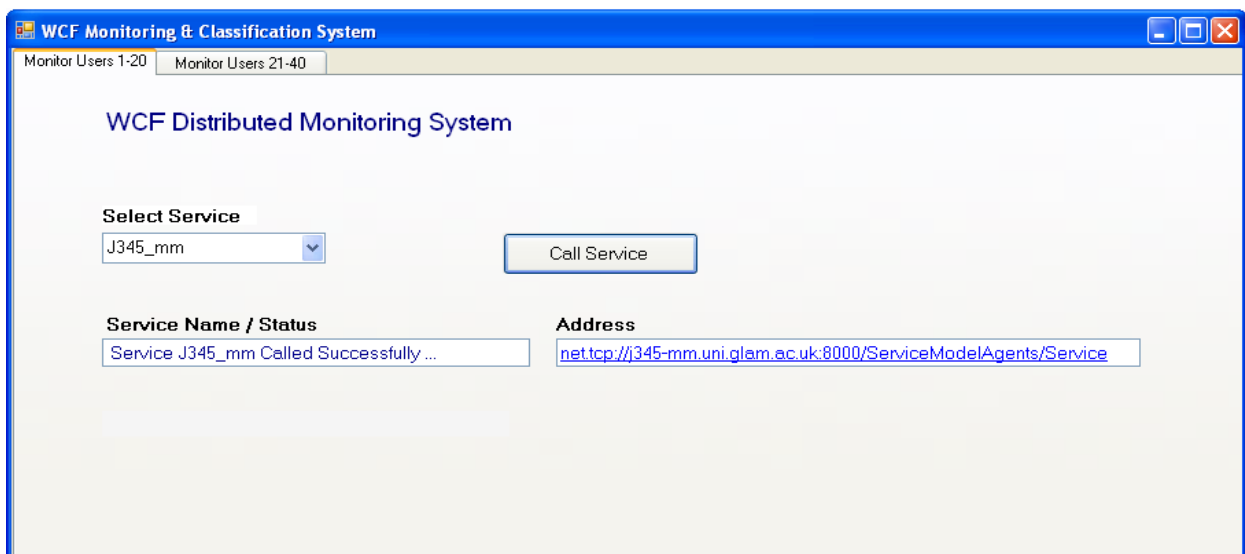


Figure 5.3 the GUI of MACS

This process can be done asynchronously for all the services exposed by the MAs in the web server application (covered in more detail in 5.2.2.1). Data collected is analysed by the DUA and stored in Oracle tables using Oracle Stored Procedures (covered in more detail in 5.2.2.3). The classification process of the data is covered in the chapter titled the Evaluation and Results. The following chapter describes in detail the implementation of the DUA agent, the SF database, and Oracle server database.

5.2.2.1 The DUA agent

The DUA is responsible for accessing the MAs in the server-side application through the WCF services they expose to retrieve the outcome of the monitoring process. However, after the WCF client has been created that includes information about the types and methods that are exposed by the services; the DUA creates an instance of the client and invokes the methods asynchronously. Making asynchronous calls is important for the client-application; this avoids blocking the GUI thread while the method call runs. Thus, it is possible for another call to be made while the current one is still running. This provides WCF applications even more flexibility to maximize throughput balanced against interactivity [102]. In order for the DUA to call the WCF services asynchronously, it creates a *Callback* function that needs to be called when the asynchronous operation is complete. The DUA then calls the service and the *Callback* function is specified in the call. Once this is done and a successful call is made, the *Callback* function executes to retrieve the result (i.e. contents of the MAs' local stores). Code 5.7 shows an asynchronous call to a service named J345_mm.

```
public void service_J345_MM()
{
    Automation_StreamTCP_MiniNetClient client = new Automation_StreamTCP_MiniNetClient();
    IAsyncResult arRetRangeArrayList = client.BeginRetRangeArrayList(RetRangeArrayListCallback, client);
}
public void RetRangeArrayListCallback(IAsyncResult ar)
{
    String[] result = ((Automation_StreamTCP_MiniNetClient)ar.AsyncState).EndRetRangeArrayList(ar);
    .....
    SendResult(result); // Send result to Oracle table
    .....
    Filtering(result); // Send result for filtering
}
```

Code 5.7 Asynchronous Callback to the service J345_mm.

All services have to be up and running in the server-side in order for the DUA to make a successful retrieval of data. If a service for example is stopped (i.e. the MA hosting the service is not running), a communication exception is thrown; this appears in the GUI as an indication of failed connection. For successful calls, the length of the variable *result* shown in code 5.7 is measured, if the string is empty, a message will appear in the GUI indicating that no records have been retrieved. If the *result* contains records, a copy of the records is stored in Oracle table using *SendResults* function that is in order to keep track of the history of development of end users. Another copy of the results is sent to the function named *Filtering* for filtering purposes. This process involves searching through the records, finding all the spreadsheets that a particular user developed during the monitoring process, and selecting the last updated version of every spreadsheet. The reason behind searching for the last updated versions is that because these spreadsheets contain all the features that end users developed over time before closing the spreadsheets. In other words, the users are expected to save their work a number of times before they finish via hitting the *Save* button. Thus, the last updated version of every spreadsheet is the most valuable one. Other measurement techniques like complexity and size of the spreadsheets could be used as alternatives to select the spreadsheets that will take part in the classification phase of the project. These techniques were dismissed because it was found that a number of spreadsheets developed by the same users are exactly the same in terms of complexity and size, as some users click the button *Save* twice or more without adding any new features to their applications. Users in some other cases may open and close their applications without making any changes; this again yields spreadsheets with the same complexity and size. Thus searching for the last updated version of every spreadsheet for every individual user has resolved the problem.

These records are stored in a different oracle table to be used in a later stage in the data mining process.

In the end of both cases of filtering and non-filtering processes, the DUA builds up the parameter sets required to use Oracle Stored Procedures and populate the parameters accordingly.

A connection to Oracle database is then made, and data is uploaded into the appropriate tables via the execute method of the connection object. Figure 5.4 shows the basic algorithm of the DUA.

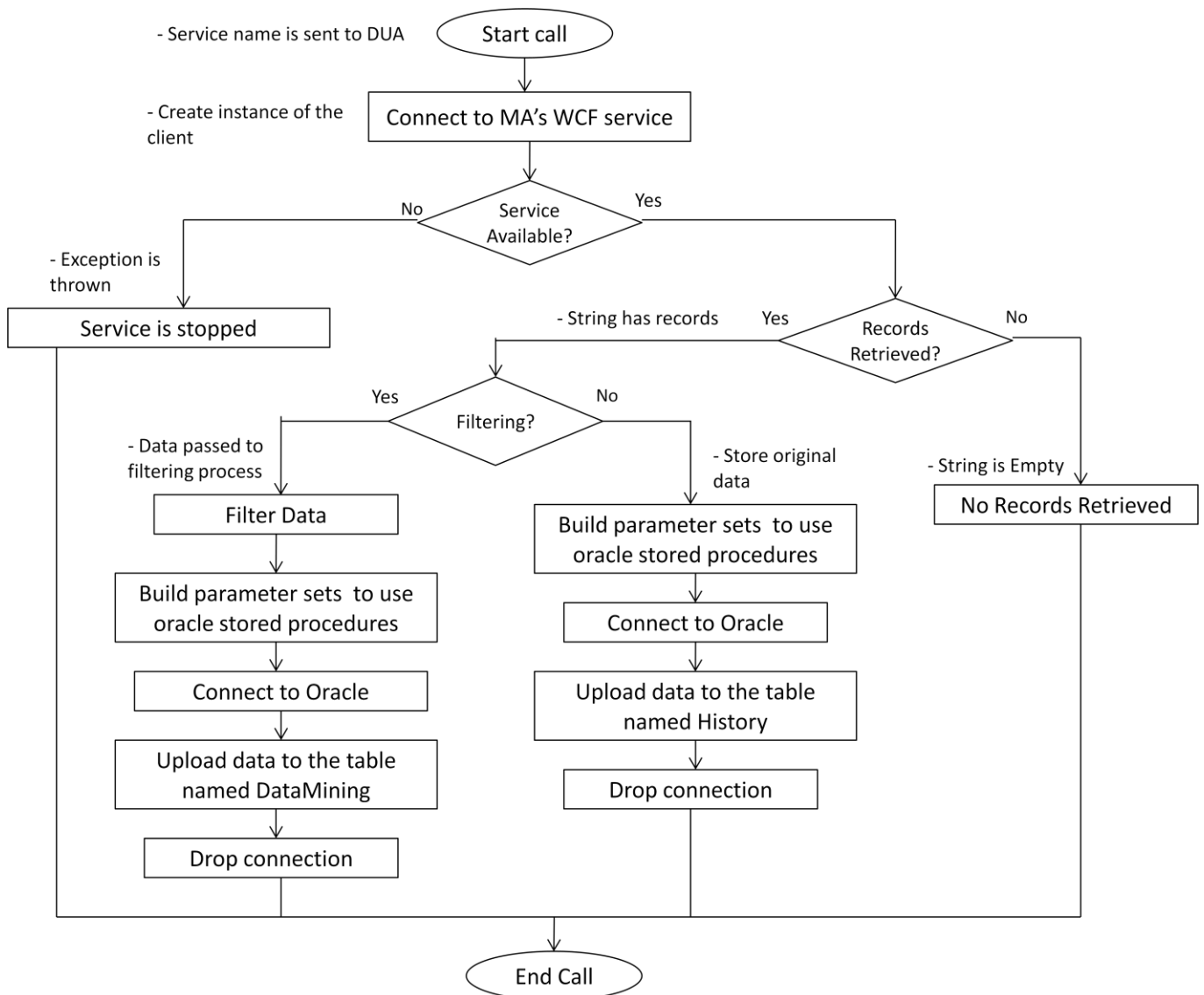


Figure 5.4 the basic algorithm of the DUA

5.2.2.2 The Service Facilitator (SF)

In FIPA specifications, the Directory Facilitator (DF) is an optional component of the MultiAgent System. In contrast, in MACS the Service Facilitator (SF) is considered as mandatory because the system is being implemented with respect to SOA architecture. The SF is being used to advertise the WCF services of the system the MAs expose. The SF database is built in Oracle. It keeps the information about the services, like the services' names, addresses and returned values. On launch of the client application, the SF responds to the UA queries requesting the services registered in the system. Code 5.8 shows querying the SF for registered services.

```

// Connect to Oracle Server Database
OracleConnection objConn = new OracleConnection("User Id=sys; Password=pword; Data Source=orl;");
OracleCommand objCmd = new OracleCommand();
OracleCommand objCmd1 = new OracleCommand();
objCmd.Connection = objConn;
objCmd1.Connection = objConn;
// Get the Numer of services available
objCmd.CommandText = "get_count_SF_tab";
objCmd.CommandType = CommandType.StoredProcedure;
objCmd.Parameters.Add("return_value", OracleType.Number).Direction = ParameterDirection.ReturnValue;
try
{
    objConn.Open();
    objCmd.ExecuteNonQuery();
    count = objCmd.Parameters["return_value"].Value;
}
catch (Exception ex)
{
    MessageBox.Show(ex.ToString()); }
try
{
    objCmd1.CommandText = "select * from Agents_Expose_Services_tab";
    OracleDataReader dr = objCmd1.ExecuteReader();
    while (dr.Read())
    {
        for (int j = 0; j < count; j++)
            collect[i, j] = dr.GetString(j);
        i += 1; }
}
catch (Exception ex)
{
    MessageBox.Show(ex.ToString()); }
    .....
    .....
objConn.Close();
objConn.Dispose();

```

Code 5.8 Querying the SF for registered services

A list of all available services is provided and appeared in a *ComboBox* area on the GUI. The UA can then instruct the DUA to communicate with the desired services to achieve the task of the data collection. The registration and unregistration of services is managed manually by the User Agent during the design stage of MACS. New services can be registered and unwanted current services can be unregistered to/from the SF as MAs are added or removed from the system. The information the SF provides allows the DUA to communicate with the desired services, instead of calling all or discarded services.

5.2.2.3 Oracle Server Database 10g Release 10.2

The data collected from the server-side application via the DUA is deposited in an Oracle Server Database. This data will form the basis for classifying the spreadsheet developers' competency using Oracle Data Mining tool (covered in more detail in the next chapter titled Evaluation and Results). However, .NET Framework Data Provider for Oracle is the technique used to upload the data into the appropriate tables. ODP.NET is used because it

enables .NET to access Oracle in a more efficient, flexible, and more stable manner than other techniques like OLE DB, and ODBC. The data is received and inserted into the tables using Oracle Stored Procedures.

5.2.3 Implementation Considerations

The development of Windows Service based agents in .NET Framework and the existence of the WCF services produced agents that are fully distributed over the network of the University of Glamorgan. The creation of the WCF services proxies added an extra advantage, is that these agents are made accessible by the client application through the WCF services they expose without worrying about how the services are delivered or where they are hosted.

Every machine on the server can have an identical version of a MA agent installed on it to achieve the task of monitoring and data collection. The existence of the Service Registry that SOA provides made it even easier for the DUA to find and communicate with the agents.

A service may be added or removed to/from the MACS by simply installing or uninstalling the corresponding MAs hosting these services. Resulting in, the added or removed services are to be registered or unregistered to/from the SF of the system that is in order to avoid calling unexist or discarded services.

The .NET Framework allows the participation in an asynchronous service call, so that services can be called while other calls are still functioning. This avoids the GUI threads of being unnecessarily blocked and adds more flexibility and interactivity to the MACS.

The .NET WCF services allowed the use of the streaming of messaging technique between the client and server applications instead of the default buffering technique. This avoided messages to timeout or to exceed the default size of messages received. Resulting in, this avoided the author of the project having to modify the configuration files via resetting the send/receive timeouts and enlarging the buffer and message sizes as messages to be received get larger over time.

The decomposition of the agents into C# classes enables the easy maintenance of a specific class without affecting the rest of the agent's code. This also provides the flexibility of introducing new classes to an agent as the service it exposes expands.

5.3 Testing of MACS

The MACS system has been tested using an exemplar spreadsheet that was developed by the author of the project. This spreadsheet was created and enhanced gradually over time via adding more advanced features in a way that satisfies all different categories of end users exist in the classification criteria. However, the server-side application has been tested to ensure that the MA agent has the capability to autonomously and continuously listen to/read the different features of Excel efficiently. The client-side application has also been tested to check the communication and data transfer processes between both sides of MACS.

5.3.1 Testing the Server-Side

However, in MACS all the Monitoring Agents that have been installed in the web servers are identical and function exactly in the same manner. Thus, two computers have been used for testing the system. The first computer represents the client machine (Master Computer) in which the DUA and GUI were held. The second computer represents the web server in which both the MA that is responsible for the monitoring process is installed and the WCF service exposed by the agent is hosted. These machines are connected together and run as a mini-network, as a communication channel has been created between them using the .NET WCF ServiceHost component provided with the address in which the agent hosting the service is located together with the supported TCP communication protocol.

A new spreadsheet named *mac33.xlsx* was created in the web server machine. Given that the monitoring agent running autonomously in the background as part of the operating system, when the spreadsheet was first launched the agent successfully managed to detect the spreadsheet and an entry was made to the local data store belongs to the web server. The spreadsheet at this stage was still blank (does not have any features created yet). Thus, the entry was recorded containing only the file and author properties like *Author, Spreadsheet Name, Date Created, and Date Last Saved...etc*, together with values of zeros to all different features of Excel the MA agent is required to monitor (see the first record of table 5.1).

After then basic functions like *SUM, AVERAGE, COUNT, DATE, and DAY* together with a bar chart were created. When the author saved the spreadsheet the agent successfully scanned it and an entry was made to the local store including the updated file and author properties together with the contents of the spreadsheet (see the second record of table 5.1). These functions were created to ensure the detection capabilities of the monitoring agent. The

complete set of formulas/functions can be found in appendix C. The spreadsheet was enhanced over time via adding more advanced features like *Date and Time Functions*, *Text Functions*, *Logical Functions*, and *Financial Functions*, and *ActiveX Controls* that was to indicate a higher category end user. Upon saving, the monitoring agent successfully scanned the document and detected the type and number of all new features and an entry was made to the local data store.

A Visual Basic Application (VBA) was added to the spreadsheet to test the agent capabilities to scan and detect the macro code constructs. A Visual Basic code containing a variety of features like *Public Functions*, *Private Function*, *Public Subroutines*, and *Private Subroutines* was taken from the Internet and placed in a VBA sheet. When the author saved the document, the monitoring agent scanned the code and utilized the *RegularExpression.cs* class for matching standards with the code constructs for string parsing and replacement (covered in detail in section 5.2.1.1). It scanned through the VBA code matching texts with patterns, that is in order to find and replace strings that take a defined format. An entry was made to the local store including the file and author properties, functions/formulas and all the detected VBA code constructs.

5.3.2 Testing the Client-Side

The client-side application initiates the connection with the web server and consumes the WCF service that its monitoring agent exposes. At first, the UA retrieved the metadata manually from the web server's WCF service and used it to create a WCF class that was used to access the monitoring agent through the endpoint of the service it exposes. A call was then made by the client machine using the DUA agent in order to collect the contents of the web server's local data store. The monitoring agent received and processed the call and data was transferred over the TCP/SOAP protocol. Data was received by the DUA agent and stored in the Oracle database.

As a result, MACS has proved successful and efficient monitoring and gathering data automatically and autonomously from end user developers, and transferring data to the client machine (*Master Machine*) for further processing. The data set collected from repeatedly monitoring the spreadsheet consists of 316 columns and over 100 rows. Therefore, a sample of 20 records and a selected number of columns is displayed in table 5.1.

Author	Spreadsheet Name	Company	Application Name	Version	Date Created	Date Last Saved	Subject
mmhereeg	mac33.xls	University of Glamorgan	Microsoft Excel	11.9999	24/11/2007 15:19	27/06/2008 12:17	Software Agents
mmhereeg	mac33.xls	University of Glamorgan	Microsoft Excel	11.9999	24/11/2007 15:19	16/09/2008 11:12	Software Agents
mmhereeg	mac33.xls	University of Glamorgan	Microsoft Excel	11.9999	24/11/2007 15:19	14/10/2008 19:39	Software Agents
mmhereeg	mac33.xls	University of Glamorgan	Microsoft Excel	11.9999	24/11/2007 14:19	29/10/2008 14:17	Software Agents
mmhereeg	mac33.xls	University of Glamorgan	Microsoft Excel	11.9999	24/11/2007 14:19	29/10/2008 14:17	Software Agents
mmhereeg	mac33.xls	University of Glamorgan	Microsoft Excel	11.9999	24/11/2007 14:19	29/10/2008 16:31	Monitoring Spreadsheet Applications
mmhereeg	mac33.xls	University of Glamorgan	Microsoft Excel	11.9999	24/11/2007 14:19	29/10/2008 16:31	Monitoring Spreadsheet Applications
mmhereeg	mac33.xls	University of Glamorgan	Microsoft Excel	11.9999	24/11/2007 14:19	29/10/2008 17:39	Monitoring Spreadsheet Applications
mmhereeg	mac33.xls	University of Glamorgan	Microsoft Excel	11.9999	24/11/2007 14:19	29/10/2008 17:39	Monitoring Spreadsheet Applications
mmhereeg	mac33.xls	University of Glamorgan	Microsoft Excel	11.9999	24/11/2007 14:19	29/10/2008 18:16	Monitoring Spreadsheet Applications
mmhereeg	mac33.xls	University of Glamorgan	Microsoft Excel	11.9999	24/11/2007 14:19	29/10/2008 18:23	Monitoring Spreadsheet Applications
mmhereeg	mac33.xls	University of Glamorgan	Microsoft Excel	11.9999	24/11/2007 14:19	29/10/2008 18:52	Monitoring Spreadsheet Applications
mmhereeg	mac33.xls	University of Glamorgan	Microsoft Excel	11.9999	24/11/2007 14:19	29/10/2008 18:52	Monitoring Spreadsheet Applications
mmhereeg	mac33.xls	University of Glamorgan	Microsoft Excel	11.9999	24/11/2007 14:19	29/10/2008 18:52	Monitoring Spreadsheet Applications
mmhereeg	mac33.xls	University of Glamorgan	Microsoft Excel	11.9999	24/11/2007 14:19	29/10/2008 18:52	Monitoring Spreadsheet Applications
mmhereeg	mac33.xls	University of Glamorgan	Microsoft Excel	11.9999	24/11/2007 14:19	29/10/2008 18:52	Monitoring Spreadsheet Applications
mmhereeg	mac33.xls	University of Glamorgan	Microsoft Excel	11.9999	24/11/2007 14:19	30/10/2008 13:22	Monitoring Spreadsheet Applications
mmhereeg	mac33.xls	University of Glamorgan	Microsoft Excel	11.9999	24/11/2007 14:19	31/10/2008 11:42	Monitoring Spreadsheet Applications
mmhereeg	mac33.xls	University of Glamorgan	Microsoft Excel	11.9999	24/11/2007 14:19	06/11/2008 21:06	Monitoring Spreadsheet Applications
mmhereeg	mac33.xls	University of Glamorgan	Microsoft Excel	11.9999	24/11/2007 14:19	24/11/2008 10:07	Monitoring Spreadsheet Applications

----- -----	Sum	Max	Min	Average	Count	Date	Day	Month	Time	---- ----	Date & Time Functions	Lookup & Ref Functions	Database Functions	Text Functions
--	0	0	0	0	0	0	0	0	0	--	0	0	0	0
--	3	0	0	4	1	3	2	0	0	--	0	0	0	0
--	3	0	0	4	1	3	2	0	0	--	0	0	0	0
--	3	0	0	4	1	3	2	1	0	--	8	0	0	6
--	3	0	0	4	1	3	2	1	0	--	8	0	0	6
--	3	0	0	4	1	4	7	1	0	--	22	0	0	6
--	4	0	0	4	1	4	7	1	0	--	22	0	0	6
--	4	0	0	4	1	4	7	1	0	--	22	0	0	6
--	4	0	0	4	1	4	7	1	0	--	22	0	0	6
--	5	0	0	4	1	5	18	1	0	--	34	0	0	6
--	5	0	0	4	1	5	18	1	0	--	34	0	0	6
--	5	0	0	4	1	5	18	1	0	--	34	1	0	7
--	5	0	0	4	1	5	18	1	0	--	34	1	0	8
--	5	0	0	4	1	5	18	1	0	--	34	2	0	8
--	5	0	0	4	1	5	18	1	0	--	34	2	0	8
--	5	0	0	4	1	5	18	1	0	--	34	2	0	8
--	5	0	0	4	1	5	18	1	0	--	34	3	0	29
--	5	0	0	11	1	5	18	1	0	--	34	3	0	29
--	5	0	0	11	1	5	18	1	0	--	34	3	0	29
--	5	0	0	12	1	5	18	1	0	--	34	3	0	29

Logical Functions	Information Functions	Financial Functions	Mathematical Functions	Statistical Functions	Cube Functions	Engineering Functions	Add-in Functions	C/C++ Functions	Chart Count
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	1
3	0	12	3	4	0	0	0	0	1
3	0	12	3	4	0	0	0	0	1
4	0	13	3	5	0	0	0	0	1
5	0	13	3	5	0	0	0	0	1
10	0	15	3	7	0	0	0	0	1
10	0	15	3	7	0	0	0	0	1
10	0	16	3	7	0	0	0	0	1
10	0	16	3	7	0	0	0	0	1
10	0	74	3	8	0	1	0	0	1
10	0	74	3	8	0	1	0	0	1
10	0	74	3	8	0	1	0	0	1
10	0	74	3	8	0	1	0	0	1
10	0	74	3	8	0	1	0	0	1
10	0	74	3	9	0	2	0	0	1
10	0	74	3	10	0	2	0	0	1
10	0	74	3	10	0	2	0	0	1
10	0	74	3	11	0	2	0	0	1

Pivot Tables	Private Functions	Public functions	Private Subroutine	Public Subroutine	As New	For Each	Do Loops	ActiveX Controls	----- -----
0	0	0	0	0	0	0	0	0	--
0	0	0	0	0	0	0	0	0	--
0	0	0	0	0	0	0	0	0	--
0	3	1	2	2	0	1	2	0	--
0	3	1	2	2	0	1	2	0	--
0	3	1	2	2	0	1	2	0	--
0	3	1	2	2	0	1	2	0	--
0	3	2	2	2	0	1	3	0	--
0	3	2	2	2	0	1	3	0	--
1	3	2	2	2	0	1	3	0	--
1	3	1	2	2	0	1	3	0	--
1	3	1	2	2	0	1	3	0	--
1	3	1	2	2	0	1	3	0	--
1	3	1	2	2	0	1	3	0	--
1	3	1	2	2	0	1	3	0	--
1	3	1	2	2	0	1	3	1	--
1	3	1	2	2	0	1	4	2	--
1	3	1	2	2	0	1	4	2	--
2	5	1	2	2	0	1	4	2	--
2	5	1	2	2	0	1	4	2	--

Table 5.1 sample of 20 records of the collected testing data

Chapter 6 - Evaluation and Results

This chapter presents the results of the classification obtained from the analysis of the data gathered by the agents of MACS. Oracle Data Mining is the tool used to undertake the task of the analysis in order to achieve the goal of the classification (explained in detail in this chapter). However, MACS is made up of two toolsets:

- Data Gathering
- Classification

The data gathering toolset consists of three parts:

- The Monitoring Agents (MAs)
- The Database Updater Agent (DUA)
- The Graphical User Interface (GUI)

The classification toolset is based on the use of Oracle Data Miner (ODMr) to generate the prediction based classification of Microsoft Excel Developers.

6.1 Data Gathering

The design and implementation of the MA agents have been described in details in chapters Four. The design and implementation of the DUA agent and the GUI have been described in chapter Five.

The MA agents collected the data by monitoring and analysing 526 excel spreadsheets developed by 80 authors. The authors were First Year students doing an Excel course as part of their studies at the University of Glamorgan / Faculty of Advanced Technology. The monitoring was done over a period of six weeks, as the students were required to undertake six exercises (One exercise a week) in order to successfully complete the course. Every analysed spreadsheet represents one record in the database with 316 attributes. These attributes contain information about the students' development behaviour (e.g. spreadsheet properties, number and type of functions used, number and type of VBA code features, and languages used, ...etc).

The majority of collected excel functions (i.e. every function represents one attribute or "column" in a data set) are grouped into categories based on Microsoft Office categorization of functions. These functions are categorized by their functionality [6], for example, the Date

and Time functions like *DATE, DATEVALUE, DAY, DAYS360, HOUR, MINUTE, MONTH...* etc are grouped and summed under one attribute called *DateAndTimeFunc* (See code segment in appendix B2). This grouping can be relied on to be robust and valid because it was well defined and established by the founder of Excel, which is Microsoft Company. Thus, there was no need to do any experiments to validate the choice of grouping. Furthermore, all different versions of Excel employ this grouping. The functions involved in the categorization and grouping process are: the financial functions, data and time functions, information functions, logical functions, maths functions, statistical functions, text functions, lookup and reference functions, database functions, cube functions, engineering functions, and add-ins and automation functions (*details of grouping can be found in appendix C*). Figure 6.1 shows the grouping used by Microsoft Excel 2007.

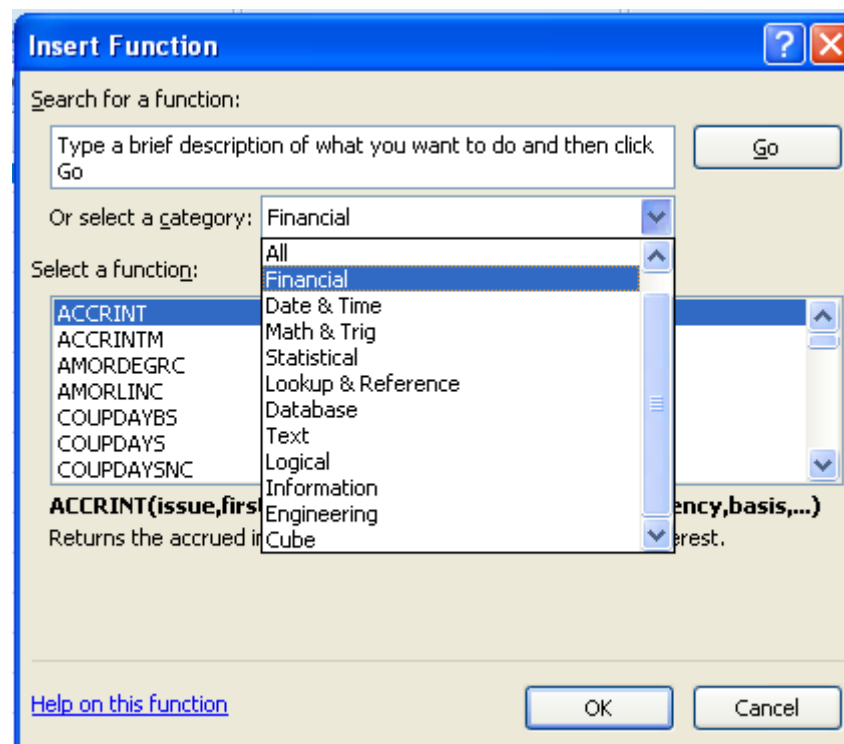


Figure 6.1 Grouping of Functions used by Microsoft Excel 2007

Table 6.1 below adopted from [81] represents a summary of this grouping. However, the number of functions involved in the modelling process reduced from 294 to 12 functions that is in addition to file properties' attributes, VBA macro code constructs' attributes and other advanced features like languages used by the users. This grouping assisted in simplifying the modelling and increasing the level of accuracy of the built model. Technically, the grouping has been done programmatically by the DUA agent during the analysis process of the gathered data.

Functions	Features
Database Functions	DAVERAGE, DCOUNT, DCOUNTA, DGET, DMAX, DMIN, DPRODUCT, DSTDEV, DSTDEVP, DSUM, DVAR, DVARP 12
Lookup & Reference Functions	ADDRESS, AREAS, CHOOSE, COLUMN, COLUMNS, HLOOKUP, HYPERLINK, INDEX, LOOKUP, MATCH, OFFSET, ROW, ROWS, TRANSPOSE, VLOOKUP, INDIRECT 16
Math & trigonometry functions	ABS, ACOS, ACOSH, ASIN, ASINH, ATAN, ATAN2, ATANH, CEILING, COMBIN, COS, COSH, DEGREES, EVEN, EXP, FACT, FLOOR, INT, LN, LOG, LOGIO, MDETERM, MINVERSE, MMULT, MOD, ODD, PI, POWER, PRODUCT, RADIANS, RAND, ROMAN, ROUND, ROUNDDOWN, ROUNDUP, SIGN, SIN, SINH, SQRT, SUBTOTAL, SUMIF, SUMPRODUCT, SUMSQ, SUMX2MY2, SUMX2PY2, SUMXMY2, TAN, TANH, TRUNC 49
Statistical Functions	AVEDEV, AVERAGEA, BETADIST, BETAINV, BINOMDIST, CHIDIST, CHIINV, CHITEST, CONFIDENCE, CORREL, COUNTA, COUNTBLANK, COVAR, CRITBINOM, DEVSQ, EXPONDIST, FDIST, FINV, FISHER, FISHERINV, FORECAST, FREQUENCY, GAMMADIST, GAMMAINV, GAMMALN, GEOMEAN, GROWTH, HARMEAN, HYGEOMDIST, INTERCEPT, KURT, LARGE, LINEST, LOGEST, LOGINV, LOGNORMDIST, MAXA, MINANEGBINOMDIST, NORMDIST, NORMEINV, PEARSON, PERCENTILE, PERCENTRANK, PERMUT, POISSON, PROB, QUARTILE, RANK, RSQ, SKEW, SLOPE, SMALL, STANDARDIZE, STDEVA, STEDP, STEDPA, STEYX, TDIST, TINV, TREND, TRIMMEANTTEST, VAR, VARA, VARP, VARPA, WEIBULL, ZTEST 67
Financial Functions	DB, DDB, SLN, SYD, VDB, AMORLINC, AMORDEGRC, CUMIPMT, CUMPRINC, EFFECT, FV, FVSCHEDULE, IPMT, IRR, ISPMT, IRR, NPV, NPV, PMT, PPMT, PV, RATE, XIRR, XNPV, COUPDAYBS, COUPDAYS, COUPDAYSNC, COUPNCD, COUPPCD, COUPNUM, CCRINT, ACCRINTM, DISC, DURATION, INTRATE, MDURATION, NOMINAL, ODPRICE, ODYIELD, ODDLPRICE, ODDLYIELD, PRICE, RICEDISC, PRICEMAT, RECEIVED, YIELD, YIELDDISC, YIELDMAT, OLLARDE, DOLLARFR, TBILLEQ, TBILLPRICE, TBILLYIELD 53
Date & Time Functions	DATE, DATEVALUE, DAY, DAYS360, HOUR, MINUTE, MONTH, NOW, SECOND, TIME, TIMEVALUE, TODAY, WEEKDAY, YEAR 14
Text Functions	CHAR, CLEAN, CODE, CONCATENATE, DOLLAR, EXACT, FIND, FIXED, LEFT, LEN, LOWER, MID, PROPER, REPLACE, REPT, RIGHT, SEARCH, TEXT SUBSTITUTE, T, TRIM, UPPER, VALUE 23
Information Functions	CELL, ERRORTYPE, INFO, ISBLANK, ISERR, ISLOGICAL, ISNA, ISNONTEXT, ISNUMBER 9

Logical Functions	AND, FALSE, IF, NOT, OR, TRUE 6
Cube Functions	CUBEKPIMEMBER, CUBEMEMBER, CUBEMEMBERPROPERTY, CUBERANKEDMEMBER, CUBESET, CUBESETCOUNT 7
Engineering Functions	BESSELI, BESSELJ, BESSELK, BESSELY, BIN2DEC, BIN2HEX, BIN2OCT, COMPLEXCONVERT, DEC2BIN, DEC2HEX, DEC2OCT, DELTA, ERF, ERF, GESTEP, HEX2BIN, HEX2DEC, HEX2OCT, IMABS, IMAGINARY, IMEXP, IMARGUMENT, IMCONJUGATE, IMCOS, IMDIV, IMLN, IMLOG10, IMLOG2, IMPOWER, IMPRODUCT, IMREAL, IMSIN, IMSQRT, IMSUB, IMSUM, OCT2BIN, OCT2DEC, OCT2HEX 38
Add-ins and Automation Functions	EUROCONVERTER, GETPIVOTDATA, REGISTER_ID, SQL_REQUEST 4

Table 6.1 Summary of the Functions' grouping

The web servers' Excel macro security environment must be altered to 'Enable all macros' that is in order to allow MAs agents to programmatically open spreadsheets to read functions and VBA macro code constructs within it. Figure 6.2 shows the alteration the Macro Settings.

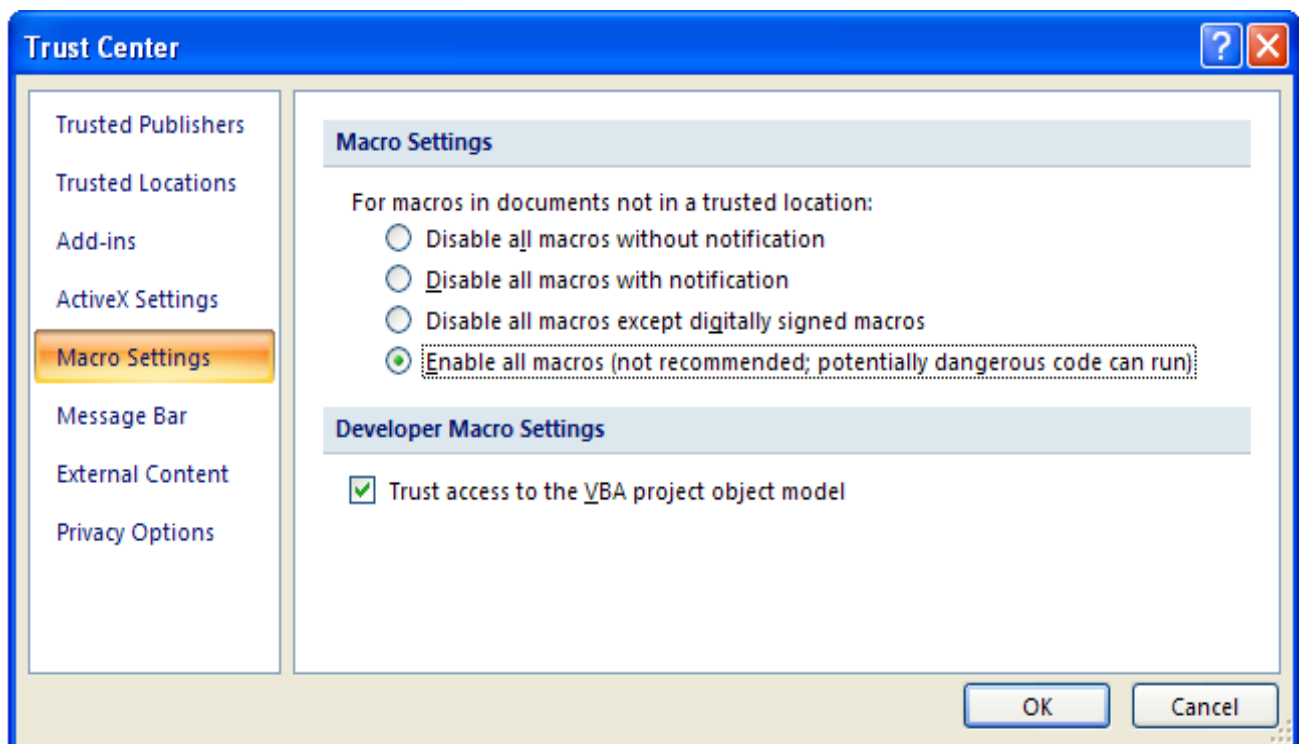


Figure 6.2 Excel Macro Security Environment Settings

6.1.1 Data Gathering Strategy

Participants were monitored for the period of six weeks. They were required to do six different Excel exercises, one exercise per week. The exercises were prepared in advance by the lecturer of the module. They were varied from basic to advanced exercises. These exercises contained a certain number/type of features and functionality the students were required to cover. All 80 students were required to do the same exercise every week. Thus, it was expected from the first week that the developed spreadsheets will be similar and contain the same sort of features. For example, students completed exercise one are expected to be novice users (i.e. Excel users as it is named in the classification criteria explained in table 6.4) as the first exercise is simple and contains features like SUM, AVERAGE, and MAX functions. Students progress to gain higher categories as they continue performing the exercises. They are expected by the author of the research to be in advanced categories once all exercises are done properly.

The Monitoring Agents have been successfully installed into the participants' machines. The agents are configured to start automatically as part of the booting process and run in the background as long as the machine is switched on. When a participant saves a spreadsheet an entry is made to the temporary data stores. When a call is received from the Database Updater Agent, data is filtered and the resultant entries are inserted into the appropriate Oracle tables. This procedure has been done in a weekly manner for all students. Data is then used by Oracle data mining algorithms for further processing that leads to the classification of these students. The agents run for the period of six weeks without any noticeable effect on the machines performance.

Table 6.2 represents a sample data for 3 students developed spreadsheets from the first week progressing to the sixth week that is to show the skill/content variation of the participants. However, cells with value 0 indicate that there was no element detected in a spreadsheet for that particular attribute (the title of the column). Cells with values greater than 0 indicate the number of times a particular element (Function, VBA code construct, Chart, Pivot Table...etc) was occurred in the spreadsheet.

In the first week students were required to cover very basic elements of excel such as *SUM*, *AVG*, and *MAX* functions. Students in the second week presented a wider understanding of excel functionality via producing spreadsheets that contained some repetitive calculations and

functions like *Text Functions*, *Logical Functions*, and *Statistical Functions*, together with occasional use of VBA macro code constructs. As the students continued performing the exercises, they managed to construct more efficient applications by making the best use of Excel functions/formulas, and making more extensive use of VBA macro code constructs.

Features weeks	Author	Spreadsheet Name	Company	Application Name	Version	Date Created	Subject
Week 1	09161015	Excel_Test_1_09161015_test_1.xls	University Of Glamorgan	Microsoft Excel	12.0	15/10/2009 11:10:22	Not Set
	09001549	Excel_Test_1_09001549_Text_1.xlsx	University Of Glamorgan	Microsoft Excel	12.0	16/10/2009 13:04:32	Not Set
	09004602	Excel_Test_1_09004602_Text_1.xlsx	University Of Glamorgan	Microsoft Excel	12.0	16/10/2009 13:05:23	Not Set
Week 2	09161015	Excel_Test_2_09161015.xlsm	University Of Glamorgan	Microsoft Excel	12.0	22/10/2009 11:07:11	Not Set
	09001549	Excel_Test_2_09001549.xlsm	University Of Glamorgan	Microsoft Excel	12.0	23/10/2009 13:09:19	Not Set
	09004602	Excel_Test_2_09004602.xlsm	University Of Glamorgan	Microsoft Excel	12.0	23/10/2009 13:11:21	Not Set
Week 3	09161015	10153.xlsm	University Of Glamorgan	Microsoft Excel	12.0	29/10/2009 11:03:48	Not Set
	09001549	309001549_3	University Of Glamorgan	Microsoft Excel	12.0	30/10/2009 13:06:22	Not Set
	09004602	46023.xlsm	University Of Glamorgan	Microsoft Excel	12.0	30/10/2009 13:11:36	Not Set
Week 4	09161015	Excel_Test_4_09161015.xlsm	University Of Glamorgan	Microsoft Excel	12.0	05/11/2009 11:18:31	Not Set
	09001549	Excel_Test_4_09001549.xlsm	University Of Glamorgan	Microsoft Excel	12.0	06/11/2009 13:03:55	Not Set
	09004602	Excel_Test_4_09004602.xlsm	University Of Glamorgan	Microsoft Excel	12.0	30/11/2009 13:07:23	Not Set
Week 5	09161015	10153.xlsm	University Of Glamorgan	Microsoft Excel	12.0	12/11/2009 11:12:36	Not Set
	09001549	15495.xls	University Of Glamorgan	Microsoft Excel	12.0	13/11/2009 13:08:52	Not Set
	09004602	46025.xlsm	University Of Glamorgan	Microsoft Excel	12.0	13/11/2009 13:13:33	Not Set
Week 6	09161015	10156.xlsm	University Of Glamorgan	Microsoft Excel	12.0	19/11/2009 11:06:21	Not Set
	09001549	15496.xlsm	University Of Glamorgan	Microsoft Excel	12.0	20/11/2009 13:08:52	Not Set
	09004602	46026.xlsm	University Of Glamorgan	Microsoft Excel	12.0	20/10/2009 13:14:02	Not Set

features weeks	Sum	Max	Min	Average	Count	Date	Day	Month	Time	Date & Time Functions	Lookup & Ref Functions	Database Functions	Text Functions
Week 1	12	1	1	0	0	2	0	0	0	2	0	0	0
	10	0	0	1	1	0	0	0	0	0	0	0	0
	10	0	0	1	0	1	0	0	1	2	0	0	0
Week 2	2	1	1	1	2	1	0	0	0	5	0	0	2
	0	0	0	1	2	1	0	0	0	5	0	0	2
	2	1	1	1	2	1	0	0	0	5	0	0	2
Week 3	2	0	1	1	2	1	0	0	0	5	0	0	2
	3	1	1	1	2	1	0	0	0	5	0	0	2
	3	1	1	1	2	1	0	0	0	5	0	0	2
Week 4	20	1	1	1	2	1	0	0	0	5	0	0	2
	4	1	1	1	2	1	0	0	0	5	0	0	2
	4	1	1	1	2	1	0	0	0	5	0	0	2
Week 5	2	0	1	1	2	1	0	0	0	5	0	0	2
	2	0	1	0	2	1	0	0	0	5	3	0	2
	2	1	1	1	0	1	0	0	0	5	3	0	2
Week 6	2	1	1	1	2	1	0	0	0	5	2	1	2
	11	1	1	1	2	1	0	0	0	5	2	1	2
	11	1	1	1	2	1	0	0	0	5	2	0	2

Features weeks	Logical Functions	Information Functions	Financial Functions	Math and Trigonometry Functions	Statistical Functions	Cube Functions	Engineering Functions	Add-in Functions	C/C++ Functions	Chart Count
Week 1	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0
Week 2	1	0	2	5	1	0	0	0	0	0
	1	0	2	1	1	0	0	0	0	0
	1	0	2	5	1	0	0	0	0	0
Week 3	1	0	2	1	1	0	0	0	0	0
	1	0	2	2	1	0	0	0	0	0
	1	0	2	2	1	0	0	0	0	0
Week 4	1	0	2	1	1	0	0	0	0	0
	1	0	2	1	1	0	0	0	0	0
	1	0	2	1	1	0	0	0	0	0
Week 5	1	0	2	1	1	0	0	0	0	0
	1	0	2	1	1	0	1	0	0	0
	1	0	2	1	1	0	1	0	0	0
Week 6	1	0	2	1	1	0	2	0	0	0
	1	0	2	1	1	0	2	0	0	0
	1	0	2	1	1	0	2	0	0	0

Features weeks	Pivot Tables	Private Functions	Public functions	Private Subroutine	Public Subroutine	As New	For Each	Do Loops	ActiveX Controls
Week 1	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0
Week 2	0	2	0	1	0	0	0	2	0
	0	2	0	1	0	0	0	2	0
	0	2	0	1	0	0	0	2	0
Week 3	0	6	1	2	4	0	1	4	0
	0	6	1	2	4	0	1	4	0
	0	6	1	2	4	0	1	4	0
Week 4	0	6	1	2	4	0	1	4	0
	0	5	1	2	2	0	1	3	0
	0	6	1	2	4	0	1	4	0
Week 5	0	6	1	2	4	0	1	4	0
	0	2	0	1	0	0	0	2	0
	0	0	0	0	0	0	0	0	0
Week 6	0	2	0	1	0	0	0	2	0
	0	2	0	1	0	0	0	2	0
	0	0	0	1	0	0	0	2	0

Table 6.2 Sample data for participants' spreadsheet development

6.2 Classification

The results from the data gathering toolset are stored in Oracle tables as explained in previous chapters. These results are then used by the Oracle Data Mining tool as input data set for further processing that leads to achieving the goal of the classification.

6.2.1 Oracle Data Mining Prediction

It is generally agreed that, Oracle data mining can improve businesses or provide solutions to business problems, all centred on helping to predict, understand and develop new insights of individual activities. However, the graphical user interface, ODMr, that is supported by Oracle Data Mining tool is utilized to analyse the results from the data gathering toolset in order to classify the excel spreadsheet developers into five different categories of: Excel User, Excel Power User, VBA Developer, Excel Developer, and Professional Excel Developer, that is according to their knowledge of excel functionality and VBA macro code constructs. The categorization is based on [16] classification criteria explained in table 6.4. This subchapter describes how ODMr was used to achieve this goal.

6.2.1.1 Algorithms for Classification

Classification is a function that ODMr provides that assigns categories or classes to the cases (records) that make up a collection of data. However, classification can be achieved via building a model based on historical data in which the target values are known. During building (training) the model, a classification algorithm finds relationships between the attributes' values (the predictors) and the target values in the build data set. These relationships are summarized in a model. The model can then be applied to a new data set (data of interest) with unknown target values in order to accurately predict the target values (classes) for each case (record) in the new data.

By default, ODMr divides the historical data used for the classification project into two data sets, one is used for building the model, and the other is for testing the model. The classification models are tested by comparing the predicted target values to known predictions in the test data set. The comparison technique is called *testing a model* which measures the predictive confidence of the model. The process of applying the classification

model to new data is called *applying the model*. The data to which the model is applied called the *apply data* or *scoring data*.

Oracle data mining supports four classification algorithms for solving classification problems: Naive Bayes, Adaptive Bayes Network, Decision Tree, and Support Vector Machine. The nature of data decides which algorithm will provide the best solution to the problem. The next subchapter describes how these algorithms can be utilized to solve the classification problem and which one provides the most accurate classification of excel developers.

6.2.1.1.1 Problem Definition

The first phase of solving a business problem using oracle data mining is that the problem must be well-defined and stated in terms of data mining functionality. The problem i am seeking to solve can be stated as *“I need to classify excel spreadsheet developers into five different categories, the prediction based classification is based on their knowledge of excel functionality and VBA macro code constructs”*. The classification results can be used by management to accurately allow for precise tailor training activities for future spreadsheet development.

Each of the 5 categories (i.e. Developer Types) is given a key value ranking from 1 to 5 for mining purposes and to distinguish each excel developer during the process of building/testing the models (covered in more detail in 6.2.1.1.3). The developer types ranking is illustrated in table 6.3.

Developer Type	Rank
Excel User	1
Excel Power User	2
VBA Developer	3
Excel Developer	4
Professional Excel Developer	5

Table 6.3 Excel Developer Types Ranking

Every developer is required to make use of certain features in order to be qualified for a certain ranking value. For example, users with ranking value 1 are expected to produce spreadsheets that contain some repetitive calculations and functions like Sum(), Average(), Count(), Max(), Min(), Date(), Time(), Day(), Month(), and some other features like storing lists, producing pivot tables and charts. Users in level 2 should present wider understanding of excel functionality using more advanced functions like Financial functions, Statistical functions, Text functions and create more complex spreadsheets for own use and help develop and debug colleagues spreadsheets, together with occasional development of VBA macro codes when appropriate. As the ranking value goes higher, developers are required to construct efficient and maintainable applications by making the best use of excel's functionality and make more extensive use of VBA macro code constructs, together with some other programming languages and applications to enhance their excel solutions. Developers with the highest ranking value of 5 are required to have knowledge of all previously mentioned features and have the capability to use other features like third party ActiveX controls, automate other applications, connect to different databases and use programming languages like C/C++ for fast custom worksheet functions, and much more. Table 6.4 summarizes the categories together with a description and the level of knowledge (Usage) required for every category.

Category	Description	Usage
Excel User	<ul style="list-style-type: none"> - Store lists - Simple repetitive calculations - Some worksheet functions - Pivot tables - Charts 	<ul style="list-style-type: none"> - Lists - =Sum(), =Average(), =Count(), =Max() , =Min(), Date(), Time(), Day(), Month(), ... - Pivot tables - Charts
Excel Power User	<ul style="list-style-type: none"> - Wide understanding of excel Functions - Create complex spreadsheets for own use and helps develop and debug colleague's spreadsheets. - Occasional use of VBA code from macro recorder or the Internet. 	<ul style="list-style-type: none"> - Date and time functions(), Financial functions(), Information functions(), Logical functions(), Maths functions(), Statistical functions(), Text functions() - Occasional VBA coding
VBA Developer	<ul style="list-style-type: none"> - Extensive use of VBA - Typically they are Power Users who started to learn VBA or Visual Basic developers who switched to VBA development - Often lack sufficient knowledge of Excel to make the best use of its features. 	<ul style="list-style-type: none"> - Extensive VBA coding - Date and time functions(), Financial functions(), Information functions(), Logical functions(), Maths functions(), Statistical functions(), Text functions()
Excel Developer	<ul style="list-style-type: none"> - Constructs efficient and maintainable applications by making the best use of excels' built-in functionality, augmented by VBA when appropriate. - Confident at developing excel based applications for both their colleagues and as part of a development team. - Constrained by their reluctance to use other programming languages and applications to augment their excel solutions. 	<ul style="list-style-type: none"> - Date and time functions(), Financial functions(), Information functions(), Logical functions(), Maths functions(), Statistical functions(), Text functions() - Cube functions(), Engineering functions(),Lookup and reference functions(), =indirect() - VBA coding when appropriate
Professional Excel Developer	<ul style="list-style-type: none"> - Design and develops excel based applications and utilities that are robust, fast, easy to use, maintainable and secure. - Excel forms the core use of their applications, but they include any other applications and languages that are appropriate 	<ul style="list-style-type: none"> - Date and time functions(), Financial functions(), Information functions(), Logical functions(), Maths functions(), Statistical functions(), Text functions() - Cube functions(),Engineering functions(),Lookup and reference functions(), =indirect()

	<p>For Example:</p> <ul style="list-style-type: none"> - They might use third party ActiveX controls - Automate other applications. - Use Windows API calls. - Use ADO to connect to external Databases. - Use C/C++ for fast custom worksheet functions (DLL/XLL add-ins in XLM macro sheets). - Use VB6 or VB.net for creating object modules and securing their code. 	<ul style="list-style-type: none"> - =EMBED(*) functions - Add-Ins & Automation functions - DATABASE Formulas - =CALL(*) , =RESULT(*), =RETURN(*) , =REGISTER(*) & UNREGISTER(*) functions - VBA coding
--	--	--

Table 6.4 Excel Developer categories

6.2.1.1.2 Data Acquisition and Preparation

Data gathered from various resources that is stored in the *SoftwareUsageAnalysisDatabase* table (i.e. Source Data) is imported into ODMr environment via the import wizard that ODM tool provides. This data is stored in a table named MINING_SOFTWAREUSAGE_BUILD and used for building the model activity and then the prediction of excel developer categories. This data set contains the 526 records that were stored by the DUA agent during the data gathering phase.

The data set is divided into two data subsets, one contains 426 records (called the *build or training data set*) that will be used for building (training) the model, and the other data subset containing the remaining 100 records will be held aside (called the *apply data set*) to which the built model will be applied in order to achieve the goal of the classification.

Since ODMr creates a profile of individual behaviour that can be applied to any individual, some specific personal identifiers like *Author_ID* and *LastSavedBy* will not take part in building the model process. However, The Attribute Importance (AI) function provided by Oracle Data Mining has been used to rank the attributes of the data set according to their significance and all the irrelevant attributes are eliminated. Thus, the accuracy and speed of the model is increased.

ODM supports both Binary and Multiclass classifications. The Multiclass classification is the selected method as the model is required to predict one of five target values for each case (the target values (classes) are stated in table 6.3), Resulting in, a new column named (*Class*) is added by the research author of the project to each record in the training data set indicating the actual target values for every record.

6.2.1.1.2.1 Attribute Importance

The data set collected from the end users consists of many attributes. According to the author's of the project experience and good understanding of the data, not all attributes will take part in building the predictive models. However, some attributes may reduce the effectiveness of the algorithm, as these columns of data attributes may contain uninformative or irrelevant information to the model. Such attributes simply add noise that can increase the size of the model and the time required for building, testing and applying the model.

Oracle data mining provides a supervised function called Attribute Importance (AI) that can be used to overcome this problem. AI uses the algorithm Minimum Description Length (MDL) to rank the attributes (predictors) according to their importance to determine the target class. Thus, this function can be utilized to measure the significance of the attributes that constitute a data set in order to answer the question “what characteristics are the best indicators for specifying the different categories of Excel spreadsheet developers.

The attribute importance supports an approach called *Feature Selection* that can be used to determine the most relevant attributes in a data set. This approach has been used to obtain the knowledge necessary to eliminate the irrelevant attributes in order to increase the speed and accuracy of the model building process.

Attribute importance is particularly useful as a pre-processor for some of the classification algorithms that were utilized to solve the classification problem of the project such as the Naïve Bayes or Support Vector Machines. The Adaptive Bayes Networks and the Decision Trees algorithms include features that rank the attributes as part of building the model process.

6.2.1.1.2.1.1 Data Preparation and Ranking

The goal is to rank the predictive attributes and identify those predictors that may have the most influence in making the predictions. The model is then built and the distinction is made between the five different categories of end users. However, a new column (attribute) called *class* is introduced to the dataset indicating the actual target values for each case (1 = Excel User, 2 = Excel Power User, 3 = VBA Developer, 4 = Excel Developer, 5 = Professional Excel Developer). Figure 6.3 presents a sample of 13 columns and 30 rows from the MINING_SOFTWAREUSAGE_BUILD dataset that was used to build the oracle data mining Attribute Importance model.

File Help												
Structure Data View Lineage												
Fetch Size: 500 Fetch Next Refresh												
CLASS	DATE AND TIME...	LOOKUP AND REFERENCE...	DATABASE...	TEXT...	LOGICAL...	INFORMATION...	FINANCIAL...	MATHEMATICAL...	STATISTICAL...	CUBE...	ENGINEERING...	ADDINS & AUTOMATION
1	0	0	0	0	0	0	0	0	0	0	0	0
1	2	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0
1	5	0	0	2	1	0	2	1	1	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0
2	5	0	0	2	1	0	2	1	1	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0
2	5	0	0	2	1	0	2	1	1	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0
3	5	0	0	2	1	0	2	1	1	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0
3	5	0	0	2	1	0	2	1	1	0	0	0
3	5	0	0	2	1	0	2	1	1	0	0	0
2	5	0	0	2	1	0	2	1	1	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0
2	5	0	0	2	1	0	2	1	1	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0
2	5	0	0	2	1	0	2	1	1	0	0	0
2	5	0	0	2	1	0	2	1	1	0	0	0
2	5	0	0	2	1	0	2	1	1	0	0	0
2	5	0	0	2	1	0	2	1	1	0	0	0
2	5	0	0	2	1	0	2	1	1	0	0	0
2	5	0	0	2	1	0	2	1	1	0	0	0
2	5	0	0	2	1	0	2	1	1	0	0	0
2	5	0	0	2	1	0	2	1	1	0	0	0
2	5	0	0	2	1	0	2	1	1	0	0	0
2	5	0	0	2	1	0	2	1	1	0	0	0
2	5	0	0	2	1	0	2	1	1	0	0	0
2	5	0	0	2	1	0	2	1	1	0	0	0

Figure 6.3 Sample build data for Attribute Importance

Figures 6.4/6.5 present the results returned by the model. 6.4 shows the histogram of the predictor attributes, and 6.5 shows the ranks of these attributes. The results show that the *Private Sub Count*, *Do Loops Count*, *Private Function Count*, *Engineering Functions*, and *Lookup and Reference Functions* have the most effect on predicting the different categories of end users. All of these predictors have scored importance of higher than 0.85/1.0. *Number of Charts*, *Month*, and *Time* have less effect on the categorization with importance of lower than 0.2/1.0. A number of attributes ranked at zero such as *Number of Pivot Tables*, *Day*, *CPP Functions*, and *Information Functions*. Others ranked at less than zero such as *Mathematical Functions*, *Date and Time Functions*, and *Date Created*. Generally, negative or zero ranked predictors do not take part in the prediction process and should be removed from the data set. These predictors indicate noise and can reduce the performance and accuracy of the built model.

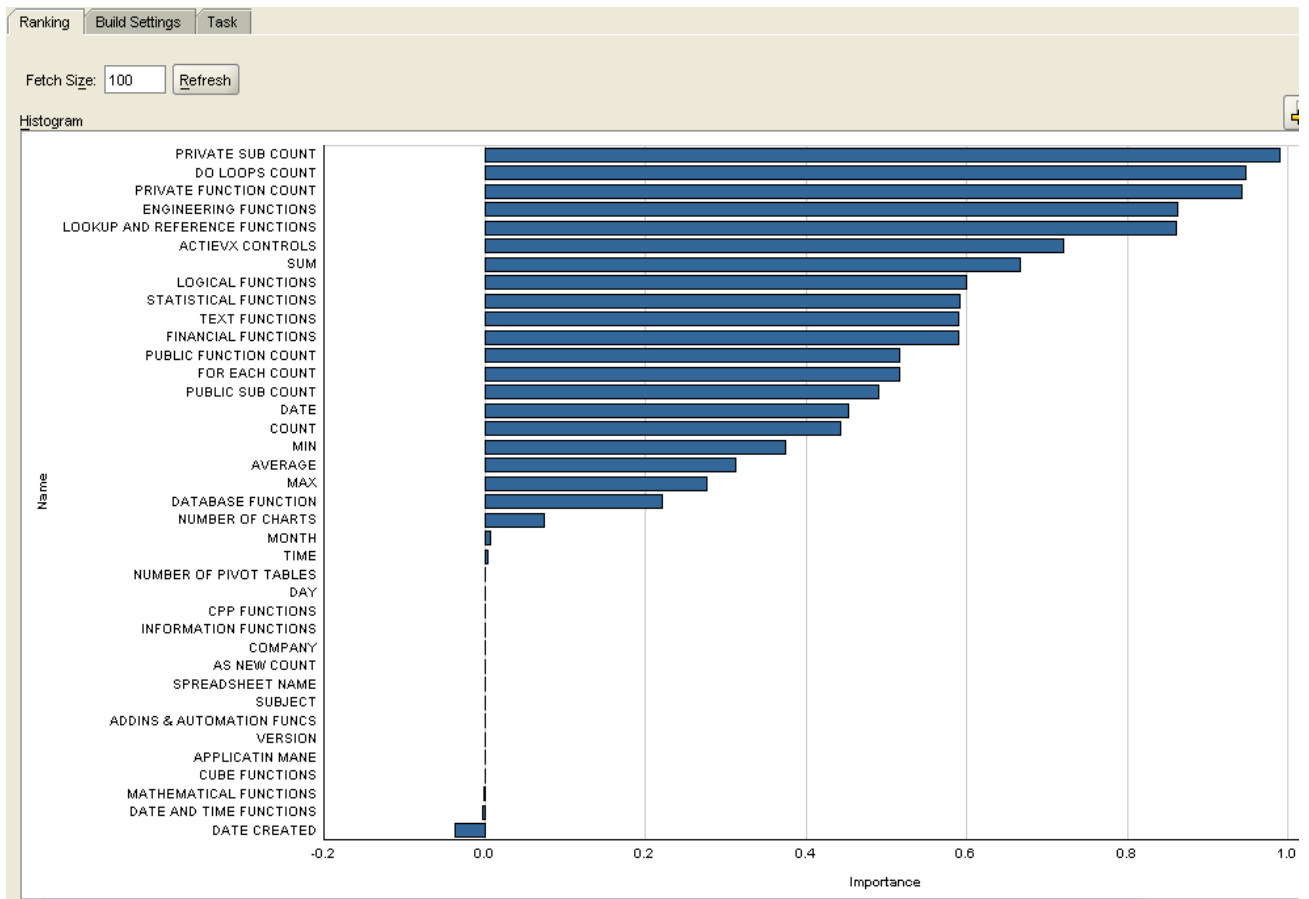


Figure 6.4 the Histogram of the predictor attributes

Ranks		
Name	Rank	Importance
PRIVATE SUB COUNT	1	0.9888530910
DO LOOPS COUNT	2	0.9468379730
PRIVATE FUNCTION COUNT	3	0.9422001970
ENGINEERING FUNCTIONS	4	0.8622732300
LOOKUP AND REFERENCE FUNCTIONS	5	0.8603101400
ACTIEVX CONTROLS	6	0.7201042420
SUM	7	0.6658302380
LOGICAL FUNCTIONS	8	0.5990754110
STATISTICAL FUNCTIONS	9	0.5917027010
TEXT FUNCTIONS	10	0.5895656870
FINANCIAL FUNCTIONS	10	0.5895656870
PUBLIC FUNCTION COUNT	11	0.5155672140
FOR EACH COUNT	11	0.5155672140
PUBLIC SUB COUNT	12	0.4894308320
DATE	13	0.4528392270
COUNT	14	0.4435310090
MIN	15	0.3751252230
AVERAGE	16	0.3119573820
MAX	17	0.2761768680
DATABASE FUNCTION	18	0.2212699220
NUMBER OF CHARTS	19	0.0737301780
MONTH	20	0.0064376640
TIME	21	0.0037817030
NUMBER OF PIVOT TABLES	22	0.0000740130
DAY	22	0.0000740130
CPP FUNCTIONS	23	0.0000000000
INFORMATION FUNCTIONS	23	0.0000000000
COMPANY	23	0.0000000000
AS NEW COUNT	23	0.0000000000
SPREADSHEET NAME	23	0.0000000000
SUBJECT	23	0.0000000000
ADDINS & AUTOMATION FUNCS	23	0.0000000000
VERSION	23	0.0000000000
APPLICATIN MANE	23	0.0000000000
CUBE FUNCTIONS	23	0.0000000000
MATHEMATICAL FUNCTIONS	24	-0.0016015380
DATE AND TIME FUNCTIONS	25	-0.0028556070
DATE CREATED	26	-0.0366029950

Figure 6.5 the ranks of the predictor attributes

An investigation was undertaken prior to deciding the removal of attributes in order to figure out why these predictors were ranked at zero or lower. The outcome was that the attribute columns *Number of Pivot Tables*, *CPP Functions*, *Information Functions*, *AsNew Count*, *Add-ins and Automation Functions*, *Cube Functions*, *Mathematical Functions*, and *Date and Time Functions* have values of zero for all cases of the dataset. This indicates that the end users did not make use of these features. Nevertheless, it has been decided by the author of the project (the expert) that these features are not to be eliminated as the data of interest's dataset may contain values for these columns. Thus, the elimination in this case may leave a negative impact on the accuracy of the model. Whereas, the attributes *Spreadsheet Name*, *Company*, *Subject*, *Version*, *Application Name*, and *Date Created* are the file properties that contain uninformative information to the model. Thus, these attributes are not contributing to the prediction as they have been removed from the data set.

After the classification problem has been defined and the sub-set of attributes that are most relevant to predicting the end user categories is defined, the model building/testing phase can now proceed.

6.2.1.1.3 Building and Testing of Models

It is generally agreed that, it is difficult to know in advance which classification algorithm will provide the best solution to the problem under consideration, normally several models are created and tested. However, in order to overcome this difficulty, all the classification algorithms that ODMr provides have been utilized to build four different models, these models are tested and compared to one another in order to pick the model that best solves the classification problem. The best model is the one that accurately predicts the classification with the highest predictive confidence.

Typically, to build a model, the build or “training” data set is split automatically by ODMr into training and testing subsets. The training subset holds 60% of the records and used to build the model, and the testing subset holds the remaining 40% of the records and used for testing the model. These default settings can be adjusted, but are recommended by the tool in order to obtain the best results.

As part of building the model activity, the built model will be applied to the test subset to compare the predictions of the model to the known data values (actual values). This comparison technique measures the model’s prediction accuracy; this ensures selecting the right model that best fits the problem.

6.2.1.1.3.1 Naïve Bayes Predictions

Naïve Bayes algorithm is the first of the Four ODMr’s classification algorithms have been used to distinguish between the five different developer types illustrated in table 6.3. This information is stored in the attribute *Class* within the build data set, each student has been assigned one of the values 1,2,3,4, or 5 to indicate the actual values of the different types of excel developers.

The Build or “Training” data set indicates what happened in the past, the model has been built based on Naïve Bayes algorithm learns from the build data set to make the distinction between different types of students. As the algorithm builds a supervised model, the model

has been applied to the test data set created in the split step during building the model. The model predictions are then compared to the actual values of the test data set. Measurements of the model effectiveness and accuracy are then explained in the Test Metrics step.

Prior to the model building, testing, and applying, ODM algorithms require that the input data set be discretized (binned). Thus, the related values of each attribute (column) in the data set were grouped together. This process reduces the number of distinct values for each attribute. Binning the data, when it is performed properly can decrease the time required to build the model, thus the models build faster. Additionally, the accuracy and confidence of the models are increased.

ODM provides three different methods for data binning, namely *TopN* binning that is used to bin categorical values. The *Equi-Width* and *Quantile* binning techniques are used to bin numerical values. The *TopN* technique was applied as the values of the dataset are of mining type *categorical*. Figure 6.6 shows how this feature was enabled.

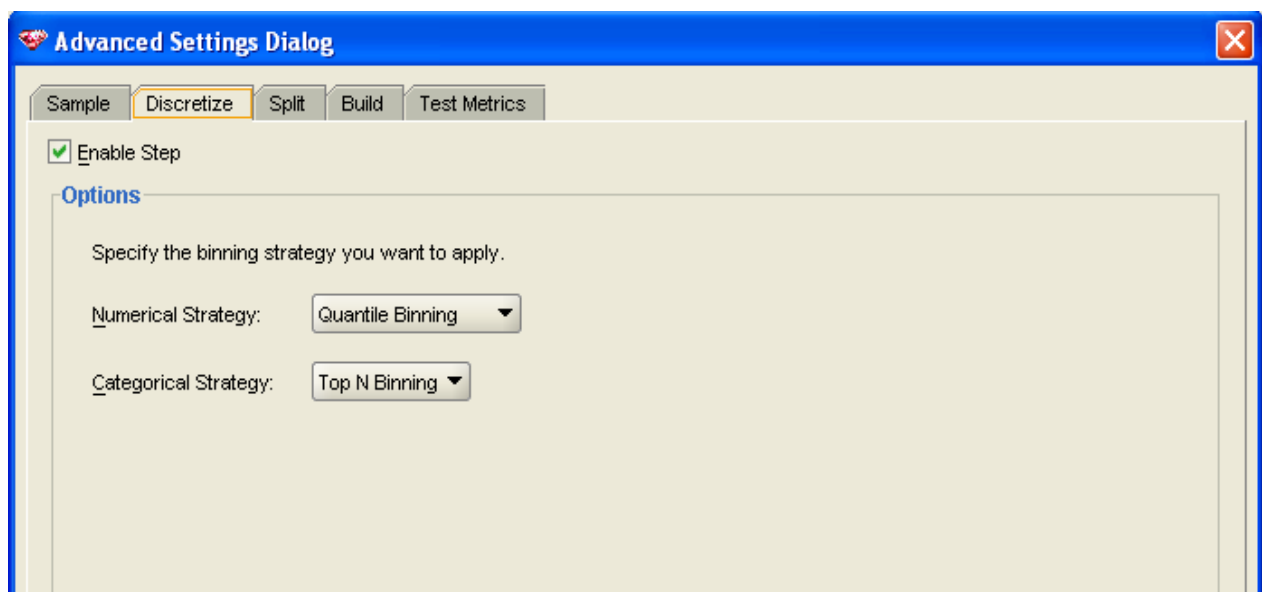


Figure 6.6 Enabling the Discretizing step

The data was binned into ranges of values based on the occurrence frequency of values, thus the data was divided into a number of bins, one for each of the values with the highest frequencies. The number of bins is decided internally by default, as it depends on the algorithm and the characteristics of the data. Some attempts have been made to perform the binning manually using the binning wizard. The accuracy was always better when applying

the default optimized settings. Figure 6.7 presents part of the binning performed as part of the Naïve Bayes model building process. The complete outcome of the binning process can be found in appendix E. The figure shows for example how the *LOOKUP AND REFERENCE FUNCTIONS* attribute was binned into 4 different groups labelled 2,3,4, an 5 based on the occurrence frequency of values within the column.

When data is binned, a probability value is also generated for each bin. Naïve Bayes uses counting techniques to calculate this feature. The probability represents the frequency percentage of the values constituting one bin within a particular attribute (column) that the bin belongs to.

Fetch Size: 426 Refresh Unbin Filter		
Attribute Name	Value	Probability
ASNEW COUNT	0	1.000000000
AVERAGE	0	0.111111111
AVERAGE	1	0.888888889
COUNT	2	0.925925925
COUNT	0	0.074074074
CPP FUNCTIONS	0	1.000000000
CUBE FUNCTIONS	0	1.000000000
DATABASE FUNCTIONS	0	1.000000000
DATE	1	1.000000000
DATE AND TIME FUNCTIONS	5	1.000000000
DAY	0	1.000000000
DO LOOPS COUNT	0	0.259259259
DO LOOPS COUNT	2	0.740740740
ENGINEERING FUNCTIONS	1	1.000000000
FINANCIAL FUNCTIONS	2	1.000000000
FOR EACH COUNT	0	1.000000000
INFORMATION FUNCTIONS	0	1.000000000
LOGICAL FUNCTIONS	1	0.814814814
LOGICAL FUNCTIONS	3	0.185185185
LOOKUP AND REFERENCE FUNCTIONS	3	0.444444444
LOOKUP AND REFERENCE FUNCTIONS	2	0.370370370
LOOKUP AND REFERENCE FUNCTIONS	5	0.037037037
LOOKUP AND REFERENCE FUNCTIONS	4	0.148148148
MATHEMATICAL FUNCTIONS	1	1.000000000
MAX	0	0.037037037
MAX	1	0.962962963

Figure 6.7 the binning of data performed for the Naïve Bayes model building.

The prediction confidence shown below is a visual indication of the effectiveness of the model and how much better the predictions made by the tested model are compared to the actual target values assigned in the build data set. The prediction confidence is normally a percentage between 0 to 100%. Figure 6.8 proves that Naïve Bayes model is 97.22% better than a naive model. In other words, the model has cut by 97.22% the errors that may occur using a naive model.

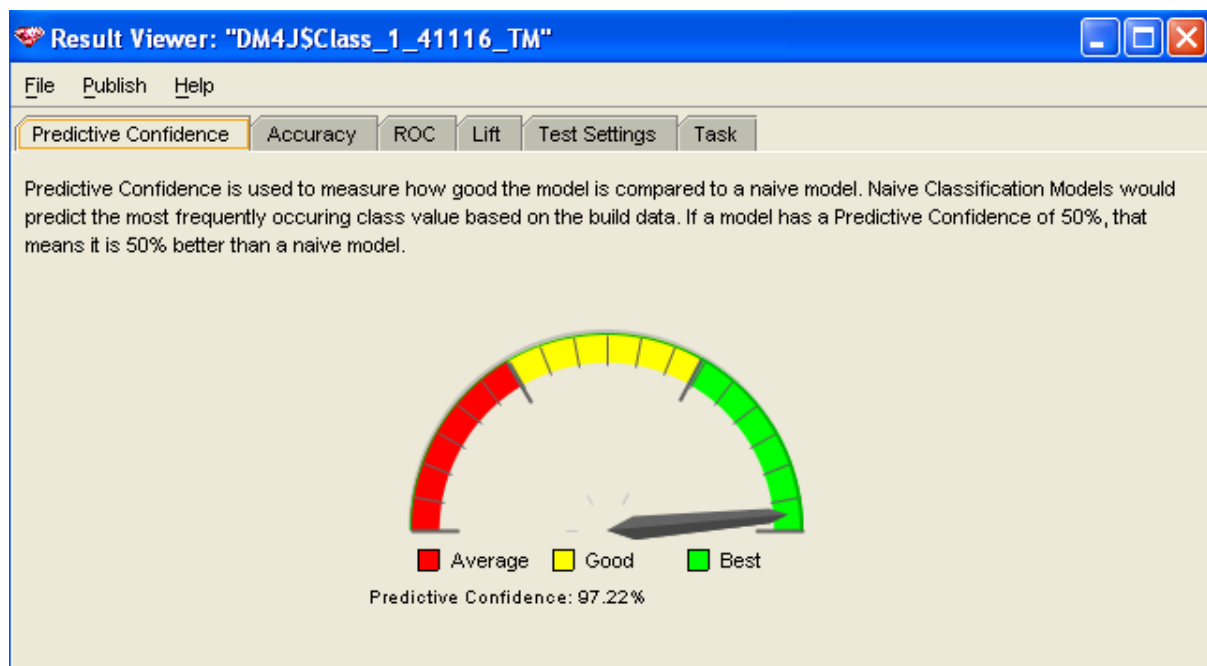


Figure 6.8 predictive confidence of Naïve Bayes algorithm

The accuracy page shown below (figure 6.9) indicates the accuracy of the model when applied to the test data set. The actual target values stated in the attribute *Class* are known, thus the predicted target values can now be compared to these values. The simplest (default) display indicates that from the 30 records with target value of 1, the model correctly predicted 96.67% of them. Likewise, the model correctly predicted 96.55% of the 29 records of target value of 2. Perfect predictions (100%) are made for the 68 and 22 records with target values 3 and 4 respectively. The model correctly predicted 95.65% of the 23 records of target values of 5.

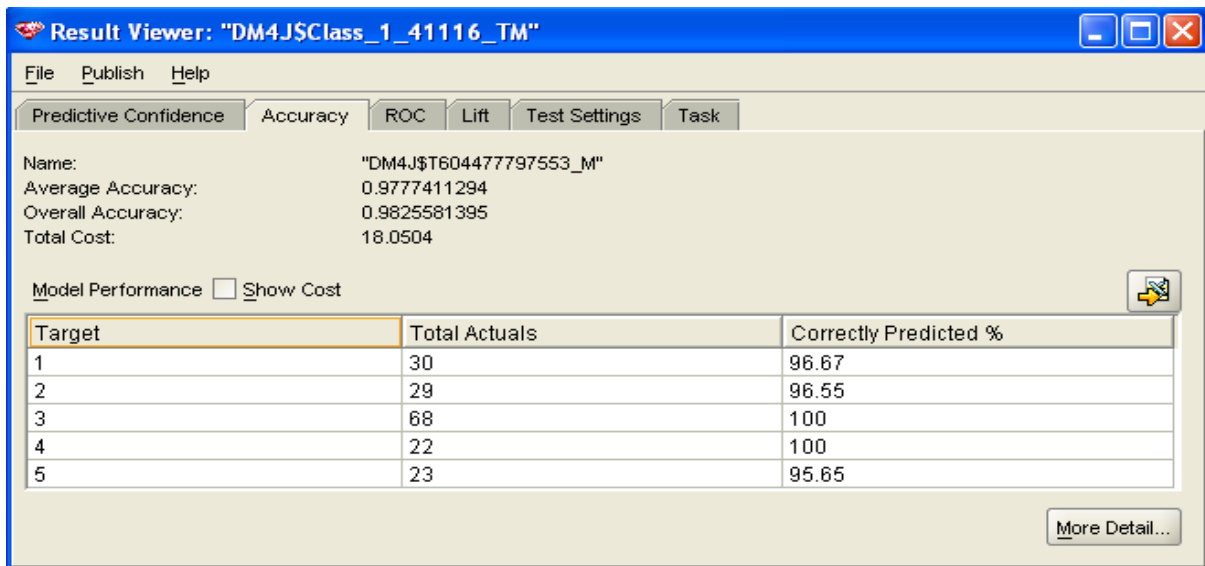


Figure 6.9 Accuracy of Naïve Bayes Algorithm

Cost is another measure in comparing one model to another. It is important to indicate the cost associated with incorrect predictions. The lower the cost is the better the model. Figure 6.10 below shows the cost of the predictions of the model.

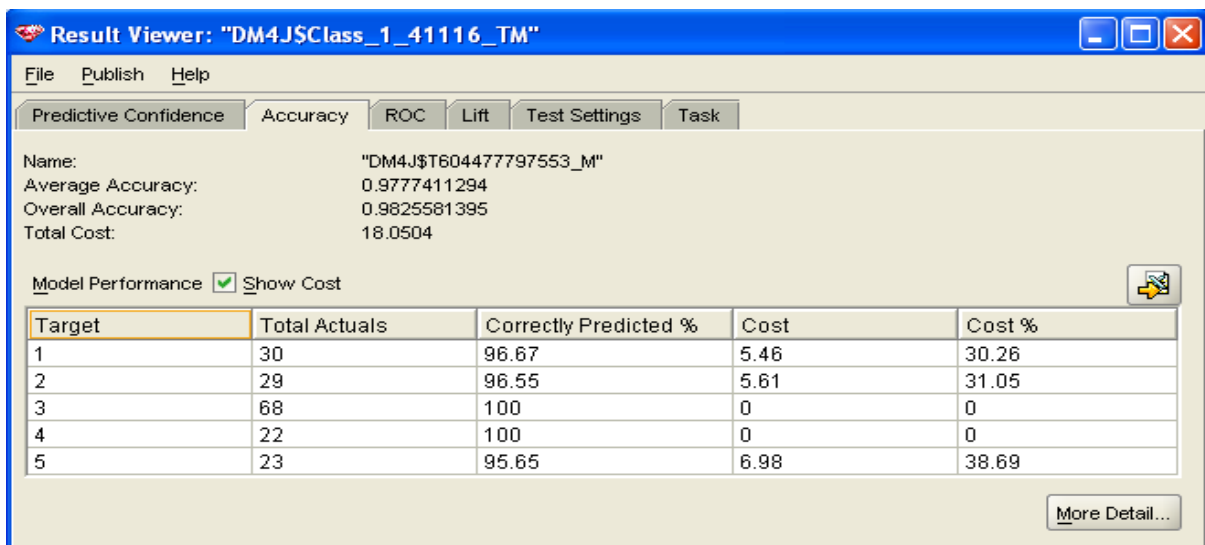


Figure 6.10 Cost of the predictions of Naïve Bayes model

The Confusion Matrix indicates the type of errors that are expected from the model. The confusion matrix is calculated by applying the model to the test data set in which the target values are already known. These target values (Classes) are compared with the predicted target values. The values of the Class attribute are known and represent the rows of the matrix, and the predictions made by the classification model represent the columns of the matrix. As shown in figure 6.11, the number 1 in row 2/colume 1 indicates a false-negative

prediction – prediction of 1 when the actual value is 2, while the number 1 in row 1/column 2 indicates a false-positive prediction – prediction of 2 when the actual value is 1.

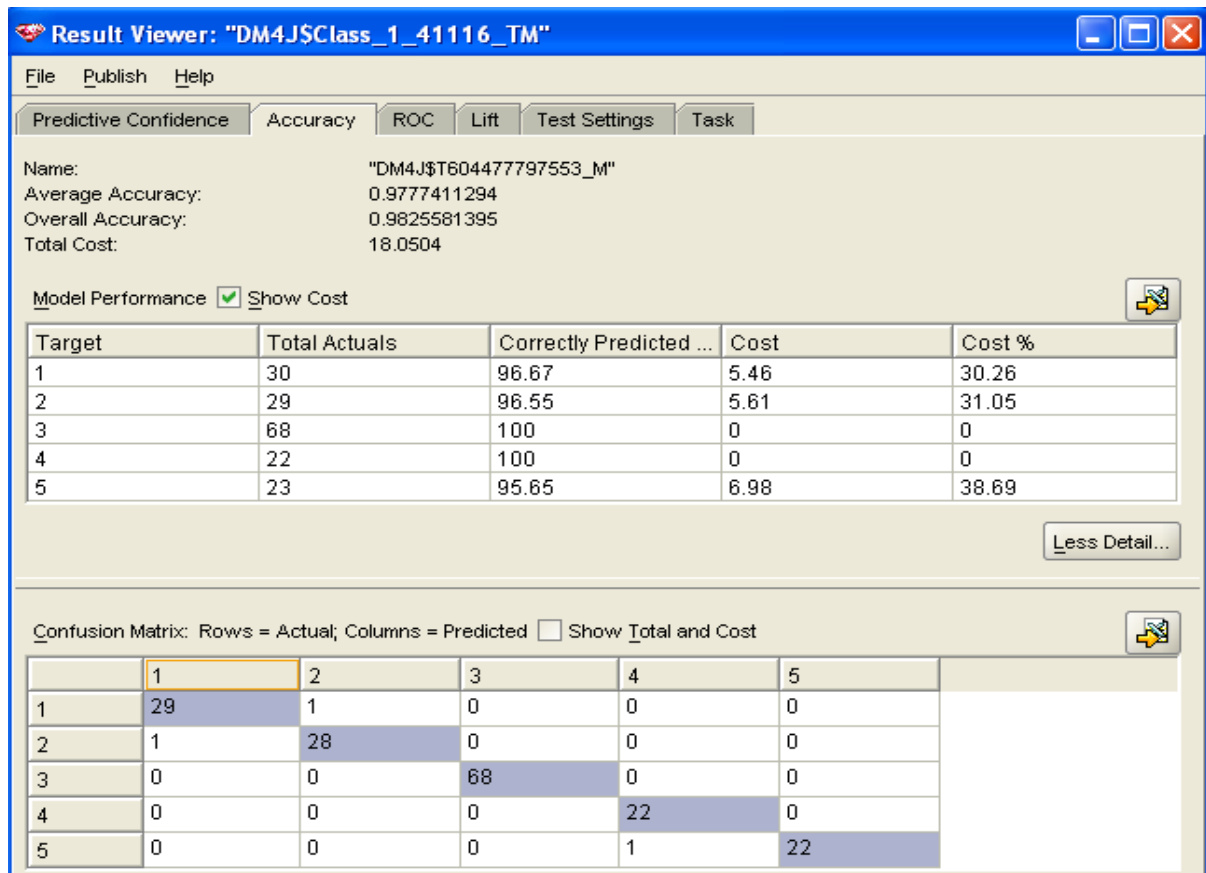


Figure 6.11 the Confusion Matrix of Naïve Bayes model

Table 6.5 below shows an analysis of the outcome of the matrix

Developer Type	Analysis of the model's Confusion Matrix
Excel User	29 of the 30 excel users have been correctly predicted. One has been incorrectly predicted as excel power user.
Excel Power User	28 out of 29 Excel Power Users have been correctly predicted. One has been incorrectly predicted as Excel User.
VBA Developer	All VBA developers have been correctly predicted.
Excel Developer	All Excel Developers have been correctly predicted.
Professional Excel Developer	22 out of the 23 Professional Excel Developers have been correctly predicted. One has been incorrectly predicted as Excel Developer.

Table 6.5 Naïve Bayes Confusion Matrix analysis

The statistics derived from the confusion matrix like the total number of cases in every category, the percentage of correctly predicted cases, and the associated costs are all displayed in figure 6.12.

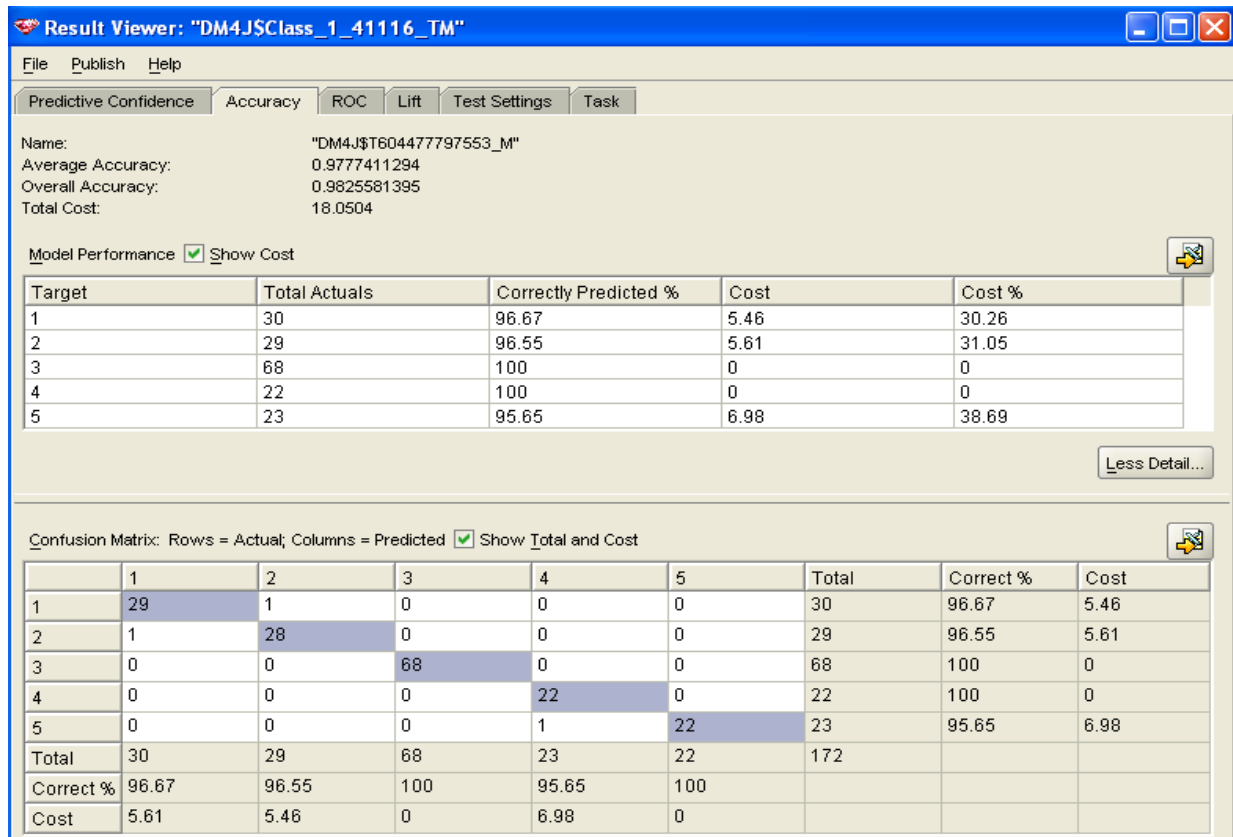


Figure 6.12 the totals and costs of Naïve Bayes Confusion Matrix

6.2.1.1.3.2 Adaptive Bayes Network Predictions

Adaptive Bayes Network algorithm is the second of the four ODMr's classification algorithms have been used for building models to distinguish between the five different developer types illustrated in table 6.3. However, using the same source data MINING_SOFTWAREUSAGE_BUILD and the same actual target value *Class*, all steps in building the model activity are identical to those of Naïve Bayes.

The prediction confidence shown below is a visual indication of the effectiveness of the Adaptive Bayes Network model and how much better the predictions made by the tested model are compared to the actual target values assigned in the build data set. Figure 6.13 proves that Adaptive Bayes Network model is 82.05% better than a naive model. In other words, the model has cut by 82.05% the errors that may occur using a naive model.

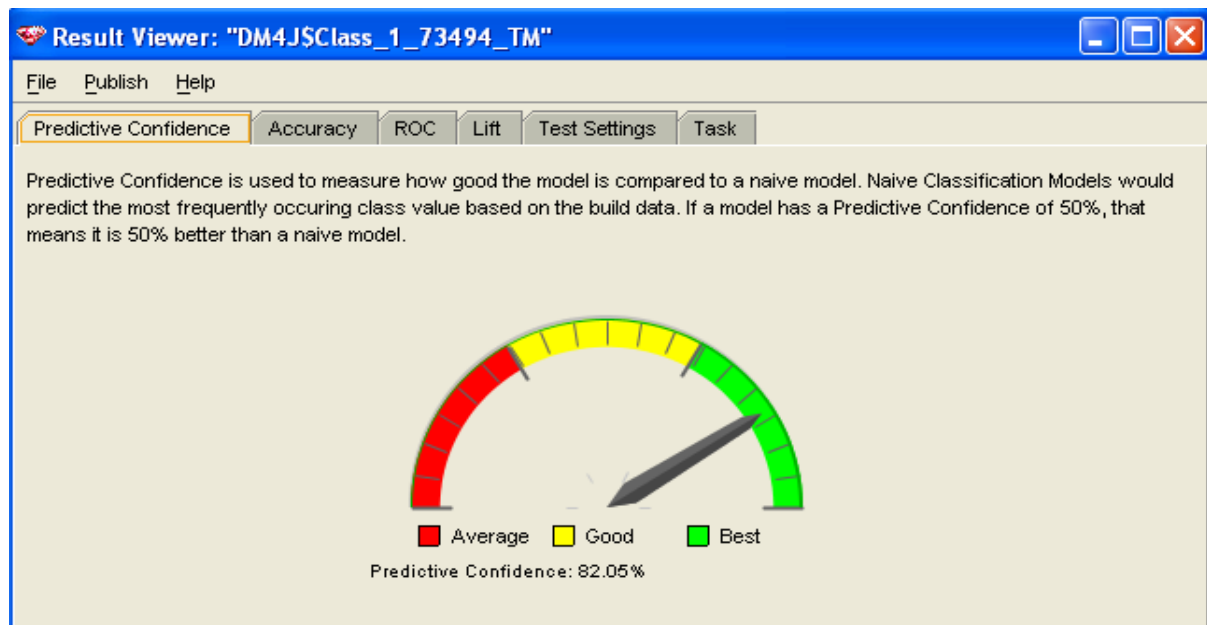


Figure 6.13 predictive confidence of Adaptive Bayes Network algorithm

The accuracy page shown below (figure 6.14) indicates the accuracy of the model when applied to the test data set. The simplest (default) display indicates that from the 30 records with target value of 1, the model correctly predicted 96.67% of them. Likewise, the model correctly predicted 100% of the 29 records of target value of 2. The model correctly predicted 75% of the 68 records of target values of 3. Perfect prediction (100%) is made for the 22 records with target values of 4. The model correctly predicted 56.52% of the 23 records of target values of 5.

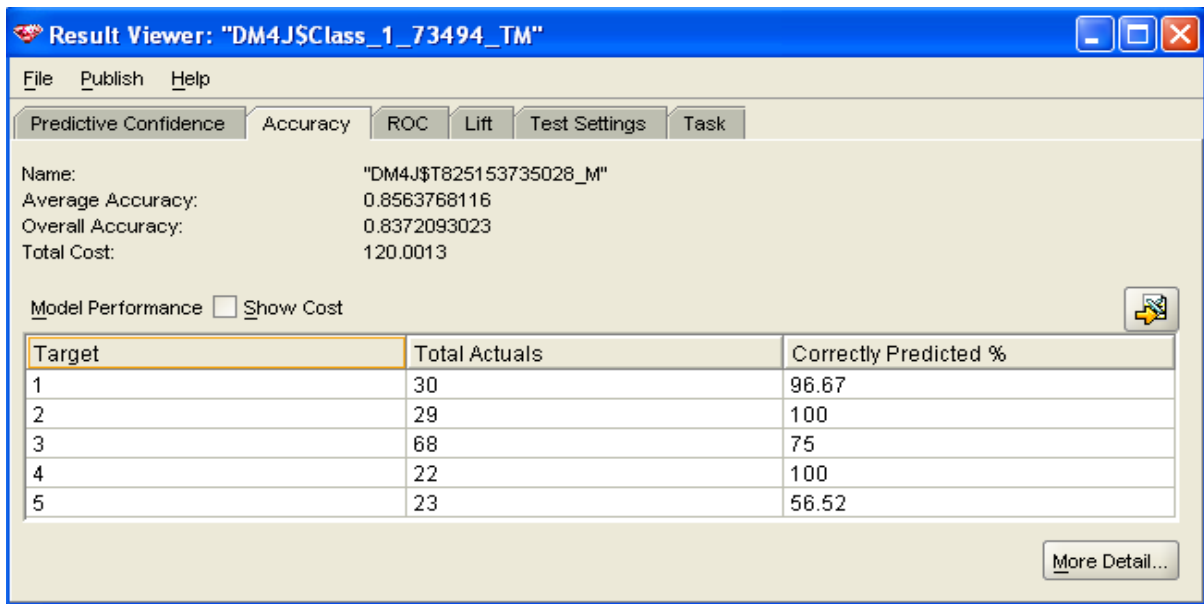


Figure 6.14 Accuracy of Adaptive Bayes Network Algorithm

Cost is another measure in comparing one model to another. It is important to indicate the cost associated with incorrect predictions. The lower the cost is the better the model. Figure 6.15 below shows the cost of the predictions of the Adaptive Bayes Network model.

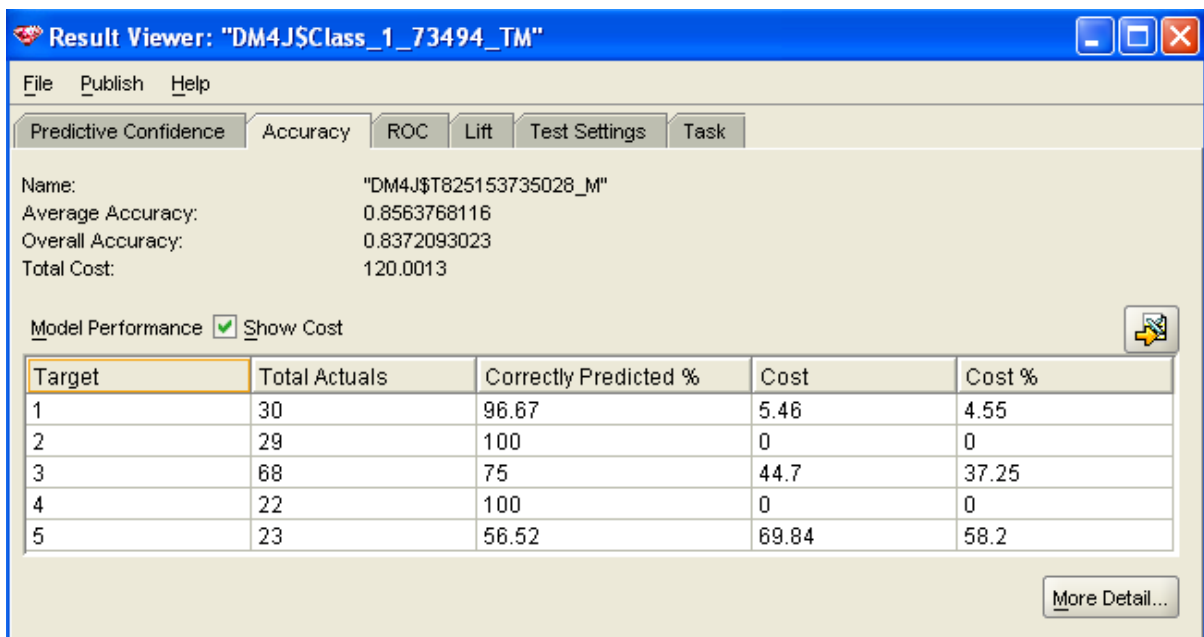


Figure 6.15 Cost of the predictions of Adaptive Bayes Network model

The Confusion Matrix indicates the type of errors that are expected from the model. The values of the Class attribute are known and represent the rows of the matrix, and the predictions made by the classification model represent the columns of the matrix. As shown in figure 6.16, the number 1 in row 1/colume2 indicates a false-positive prediction –

prediction of 2 when the actual value is 1, while the number 17 in row 3/column 1 indicates a false-negative prediction – prediction of 1 when the actual value is 3.

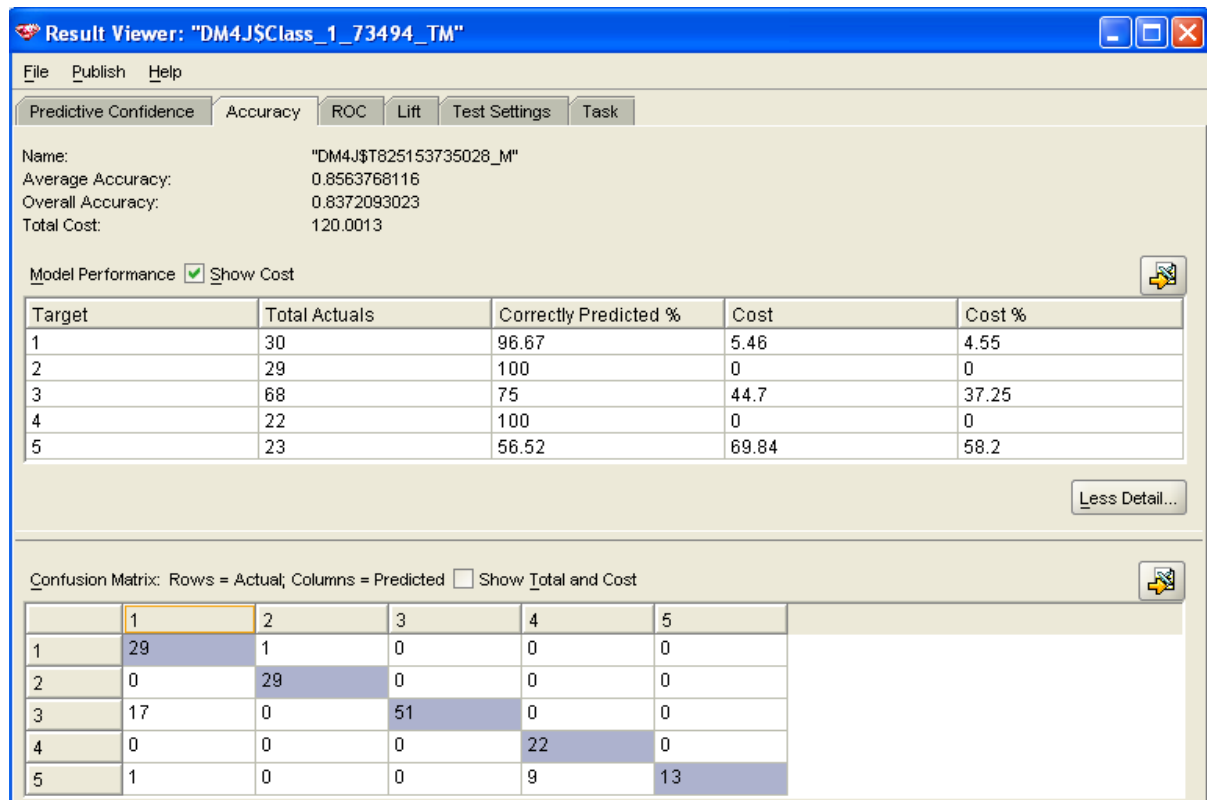


Figure 6.16 the Confusion Matrix of Adaptive Bayes Network model

Table 6.6 below shows an analysis of the outcome of the matrix

Developer Type	Analysis of the model's Confusion Matrix
Excel User	29 of the 30 excel users have been correctly predicted. One has been incorrectly predicted as Excel Power User.
Excel Power User	All the Excel Power Users have been correctly predicted.
VBA Developer	51 out of the 68 VBA Developers have been correctly predicted. 17 have been incorrectly predicted as Excel Users.
Excel Developer	All Excel Developers have been correctly predicted.
Professional Excel Developer	13 out of the 23 Professional Excel Developers have been correctly predicted. One has been incorrectly predicted as Excel User, and 9 incorrectly predicted as Excel Developers.

Table 6.6 Adaptive Bayes Network Confusion Matrix analysis

The statistics derived from the confusion matrix like the total number of cases in every category, the percentage of correctly predicted cases, and the associated costs are all displayed in figure 6.17.

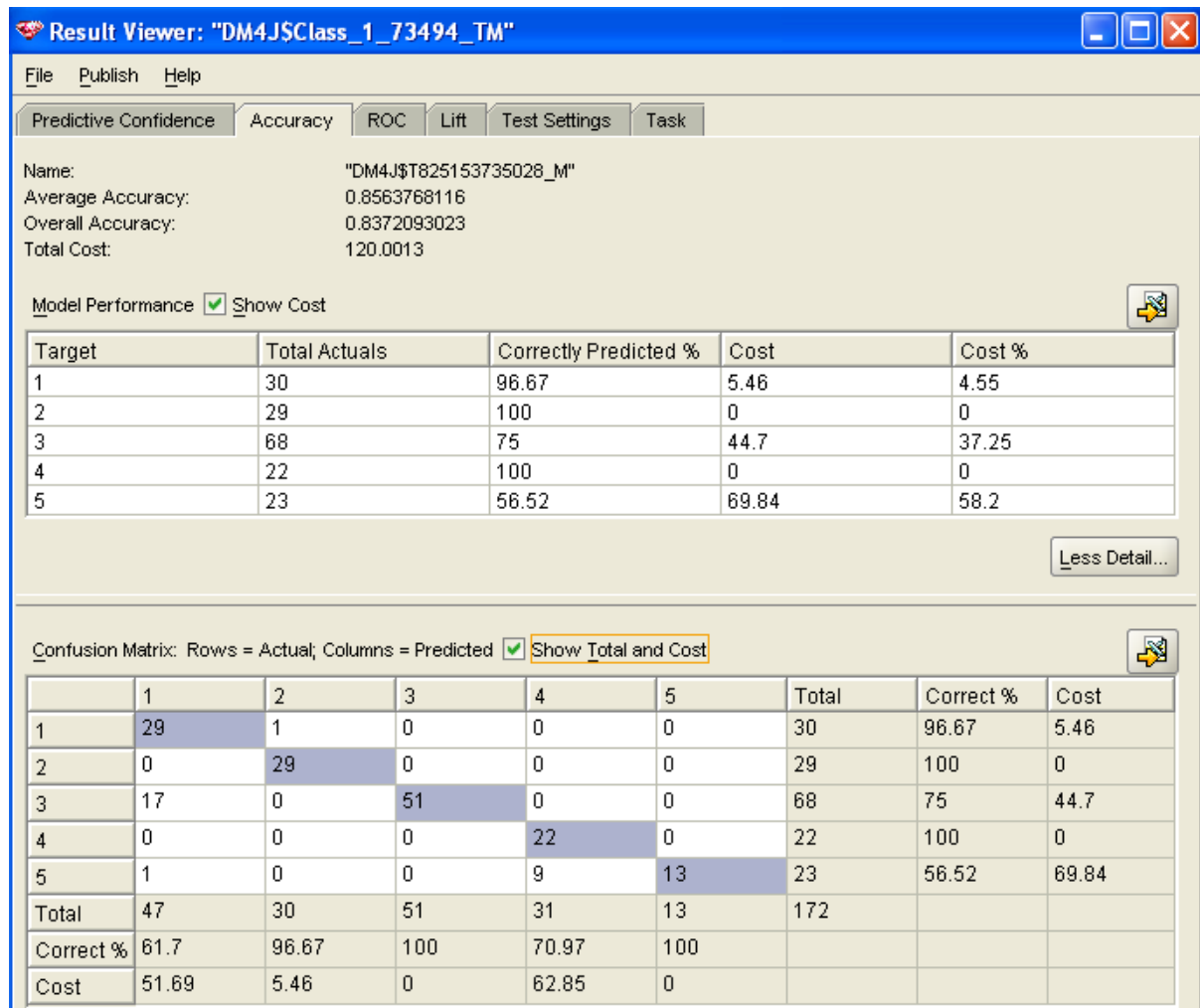


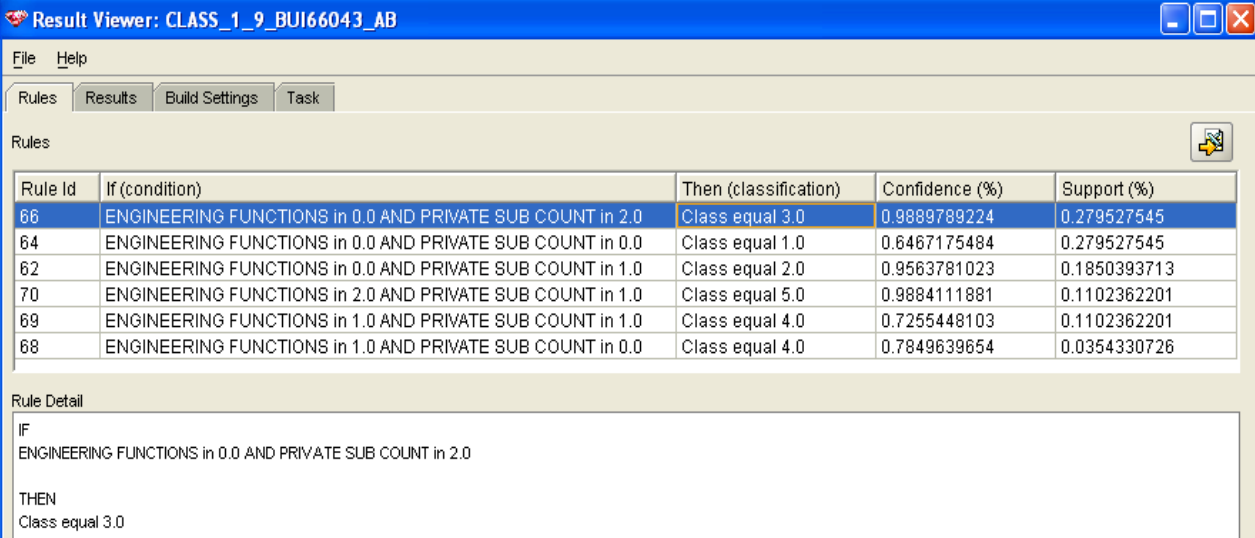
Figure 6.17 the totals and costs of Adaptive Bayes Network Confusion Matrix

ABN has the capability to generate rules. This is considered as one of the advantages of the algorithm. These rules are provided in a human-understandable form, so that the business uses can understand the basis of the predictions of the model and therefore, explain them to others. However, as explained in 2.2.3.2.3 ABN supports three different choices for Model Types, Single-Feature Build, Naive Bayes Build, and Multi-Feature Build.

Generally, users have the choice to select the ABN model type. The default Single-Feature Build type was chosen for building the model. The algorithm started with ranking the

attribute values using the Attribute Importance feature that ODM provides. This feature is included in the ABN to run as part of the model building process. The use of this feature enabled the author of the project (expert) to select the attributes that are most relevant and have positive influence on building the model. Naive Bayes is used in this type as a baseline for building the model, and the number of attributes involved in the build process taken in order from the ranked list.

The rules are generated during the building step of the model together with the confidence and support features. The confidence represents the conditional probability of the prediction, while the support represents how frequently the items involved in the rule occur together. ABN rules are in the form ‘IF prediction information THEN target’. Figure 6.18 shows the rules generated by the model. The *Rule Id 66* for example indicates that IF *ENGINEERING FUNCTION* is 0 AND *PRIVATE SUB COUNT* is 2 THEN the user is VBA Developer (class equal 3) with confidence of 0.9889789224 and support of 0.279527545. Since the source data is a small data set (426 records), the rules generated are very simple. Furthermore, it has been agreed that it is not expected from ABN algorithm to generate meaningful rules unless the source data is much larger, for instance over 20,000 records [31].



Rule Id	If (condition)	Then (classification)	Confidence (%)	Support (%)
66	ENGINEERING FUNCTIONS in 0.0 AND PRIVATE SUB COUNT in 2.0	Class equal 3.0	0.9889789224	0.279527545
64	ENGINEERING FUNCTIONS in 0.0 AND PRIVATE SUB COUNT in 0.0	Class equal 1.0	0.6467175484	0.279527545
62	ENGINEERING FUNCTIONS in 0.0 AND PRIVATE SUB COUNT in 1.0	Class equal 2.0	0.9563781023	0.1850393713
70	ENGINEERING FUNCTIONS in 2.0 AND PRIVATE SUB COUNT in 1.0	Class equal 5.0	0.9884111881	0.1102362201
69	ENGINEERING FUNCTIONS in 1.0 AND PRIVATE SUB COUNT in 1.0	Class equal 4.0	0.7255448103	0.1102362201
68	ENGINEERING FUNCTIONS in 1.0 AND PRIVATE SUB COUNT in 0.0	Class equal 4.0	0.7849639654	0.0354330726

Rule Detail

IF
ENGINEERING FUNCTIONS in 0.0 AND PRIVATE SUB COUNT in 2.0

THEN
Class equal 3.0

Figure 6.18 Rules generated for Single-Feature Build.

6.2.1.1.3.3 Decision Trees Predictions

Decision Tree algorithm is the third of the four ODMr's Classification algorithms have been used for building models to distinguish between the five different developer types illustrated in table 6.3. However, using the same source data MINING_SOFTWAREUSAGE_BUILD and the same actual target value *Class*, all steps in building the model activity are identical to those of Naïve Bayes.

The prediction confidence shown below is a visual indication of the effectiveness of the Decision Tree model and how much better the predictions made by the tested model are compared to the actual target values assigned in the build data set. Figure 6.19 proves that Decision Tree model is 96.36% better than a naive model. In other words, the model has cut by 96.36% the errors that may occur using a naive model.

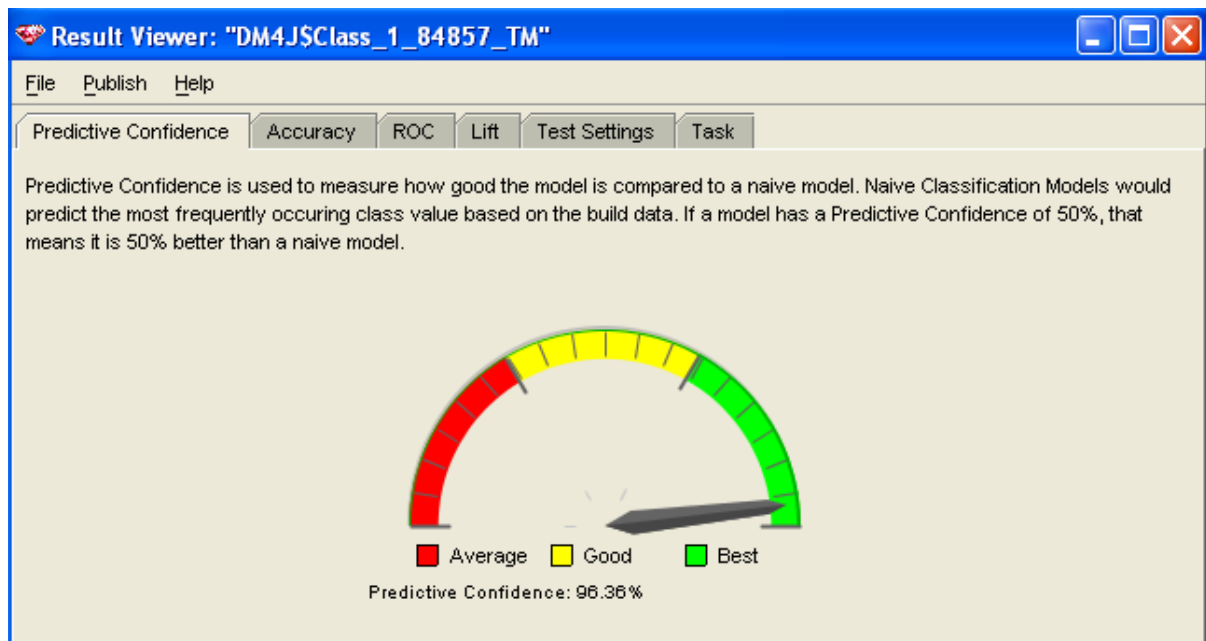


Figure 6.19 predictive confidence of Decision Tree algorithm

The accuracy page shown below (figure 6.20) indicates the accuracy of the model when applied to the test data set. The simplest (default) display indicates that from the 30 records with target value of 1, the model correctly predicted 96.67% of them. Likewise, the model correctly predicted 93.1% of the 29 records of target value of 2. Perfect predictions (100%) are made for the 68 and 22 records with target values 3 and 4 respectively. The model correctly predicted 95.65% of the 23 records of target value of 5.

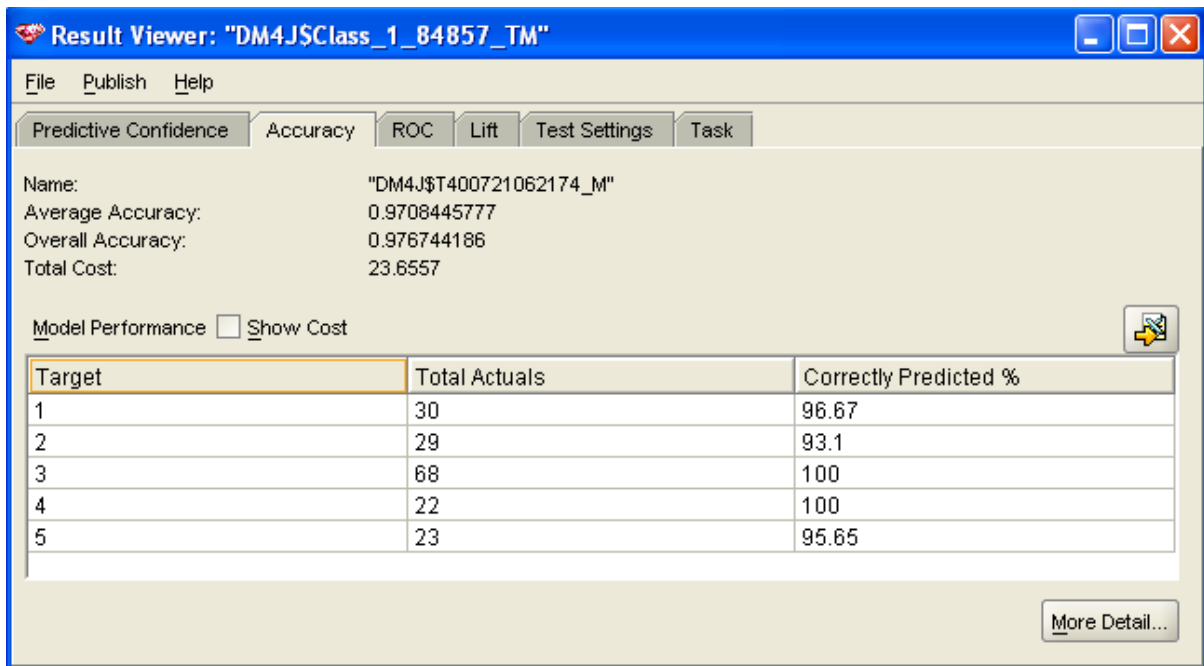


Figure 6.20 Accuracy of Decision Tree Algorithm

Cost is another measure in comparing one model to another. It is important to indicate the cost associated with incorrect predictions. The lower the cost is the better the model. Figure 6.21 below shows the cost of the predictions of the Decision Tree model.

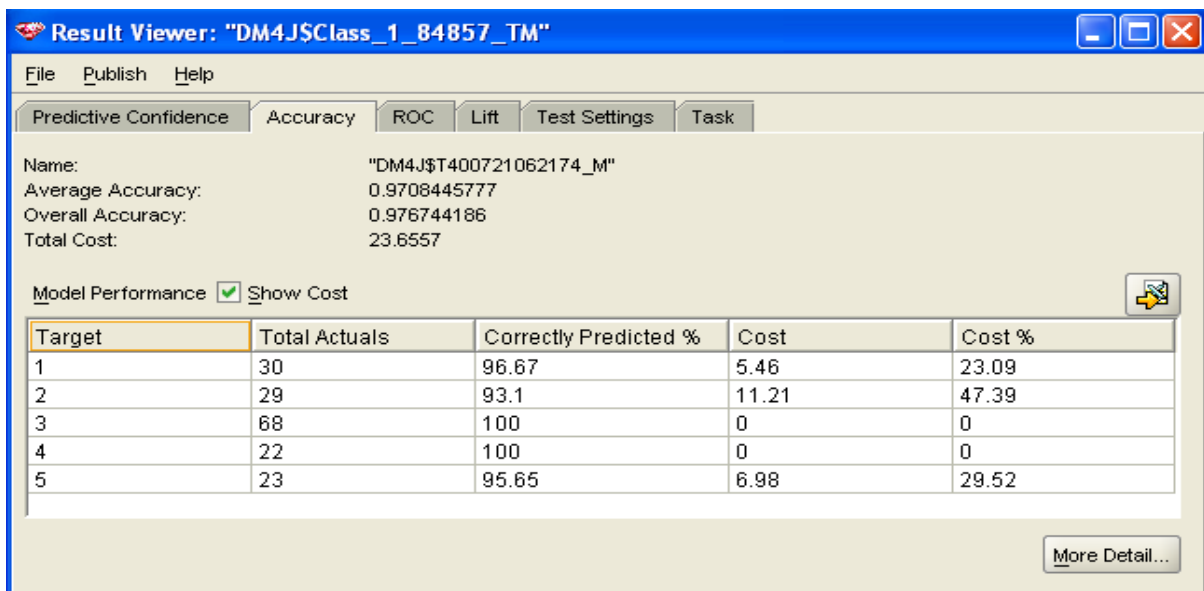


Figure 6.21 Cost of the predictions of Decision Tree model

The Confusion Matrix indicates the type of errors that are expected from the model. The values of the *Class* attribute are known and represent the rows of the matrix, and the predictions made by the classification model represent the columns of the matrix. As shown in figure 6.22, the number 1 in row 1/colume2 indicates a false-positive prediction –

prediction of 2 when the actual value is 1, while the number 1 in row 5/column 4 indicates a false-negative prediction – prediction of 4 when the actual value is 5.

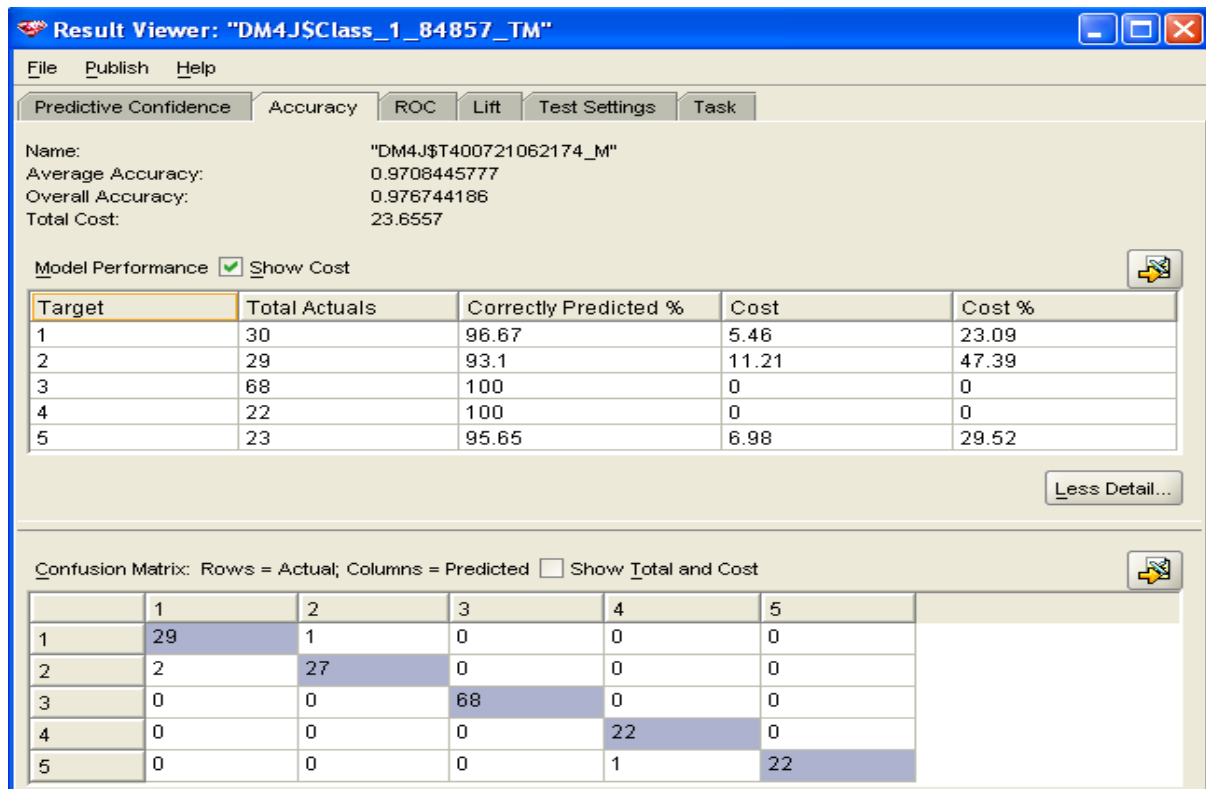


Figure 6.22 the Confusion Matrix of Decision Tree model

Table 6.7 below shown an analysis of the outcome of the matrix

Developer Type	Analysis of the model's Confusion Matrix
Excel User	29 of the 30 Excel Users have been correctly predicted. One has been incorrectly predicted as Excel Power User.
Excel Power User	27 of the 29 Excel Power Users have been correctly predicted. Two have been incorrectly predicted as Excel Users.
VBA Developer	All VBA Developers have been correctly predicted.
Excel Developer	All Excel Developers have been correctly predicted.
Professional Excel Developer	22 out of the 23 Professional Excel Developers have been correctly predicted. One has been incorrectly predicted as Excel Developer.

Table 6.7 Decision Trees Confusion Matrix analysis

The statistics derived from the confusion matrix like the total number of cases in every category, the percentage of correctly predicted cases, and the associated costs are all displayed in figure 6.23.

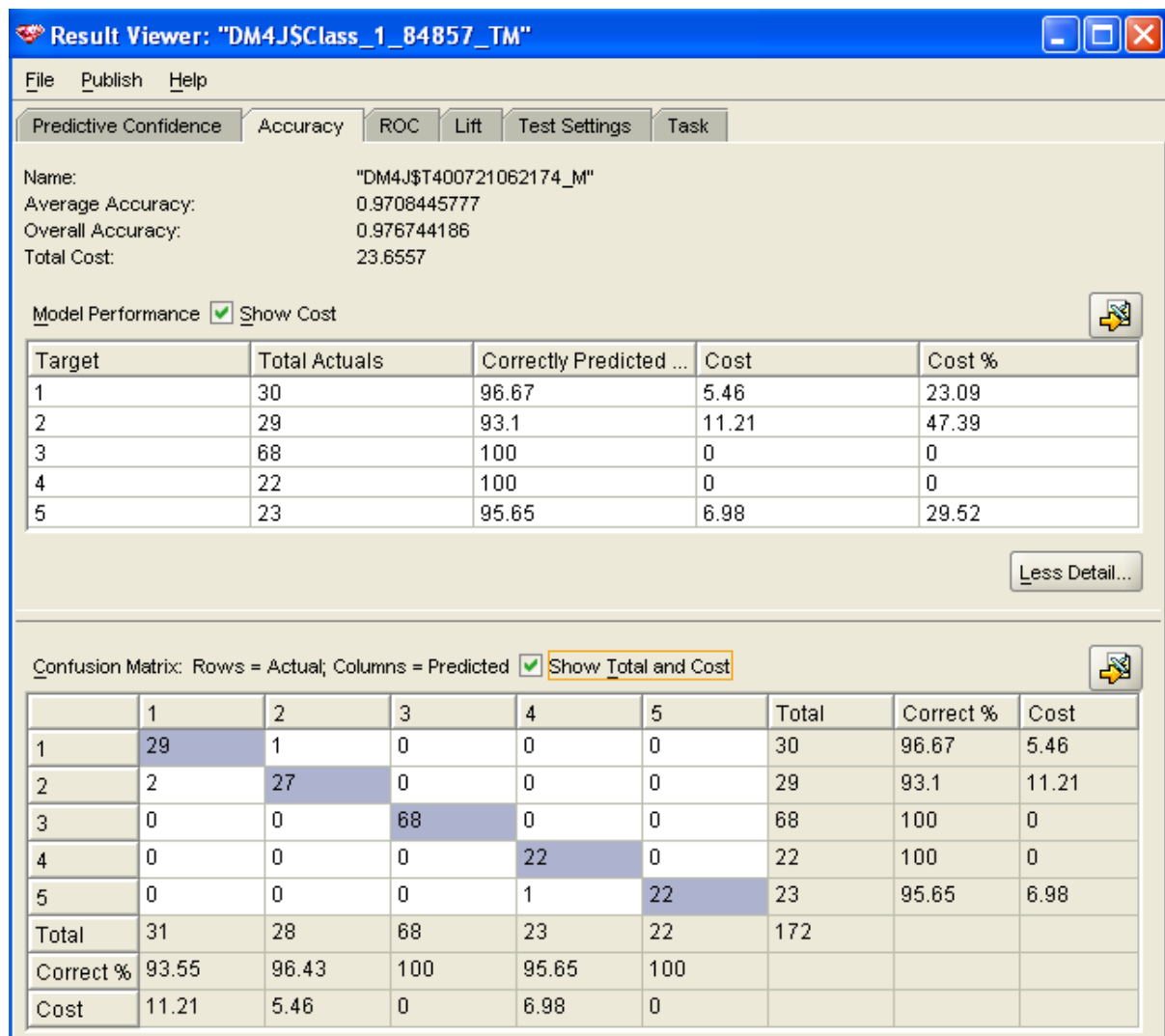


Figure 6.23 the totals and costs of Decision Tree Confusion Matrix

6.2.1.1.3.4 Support Vector Machine Predictions

Support Vector Machine algorithm is the fourth of the four ODMr's classification algorithms have been used for building models to distinguish between the five different developer types illustrated in table 6.3. However, using the same source data MINING_SOFTWAREUSAGE_BUILD and the same actual target value *Class*, all steps in building the model activity are identical to those of Naïve Bayes.

The prediction confidence shown below is a visual indication of the effectiveness of the Support Vector Machine model and how much better the predictions made by the tested model are compared to the actual target values assigned in the build data set. Figure 6.24 proves that Support Vector Machine model is 92.73% better than a naive model. In other words, the model has cut by 92.73% the errors that may occur using a naive model.

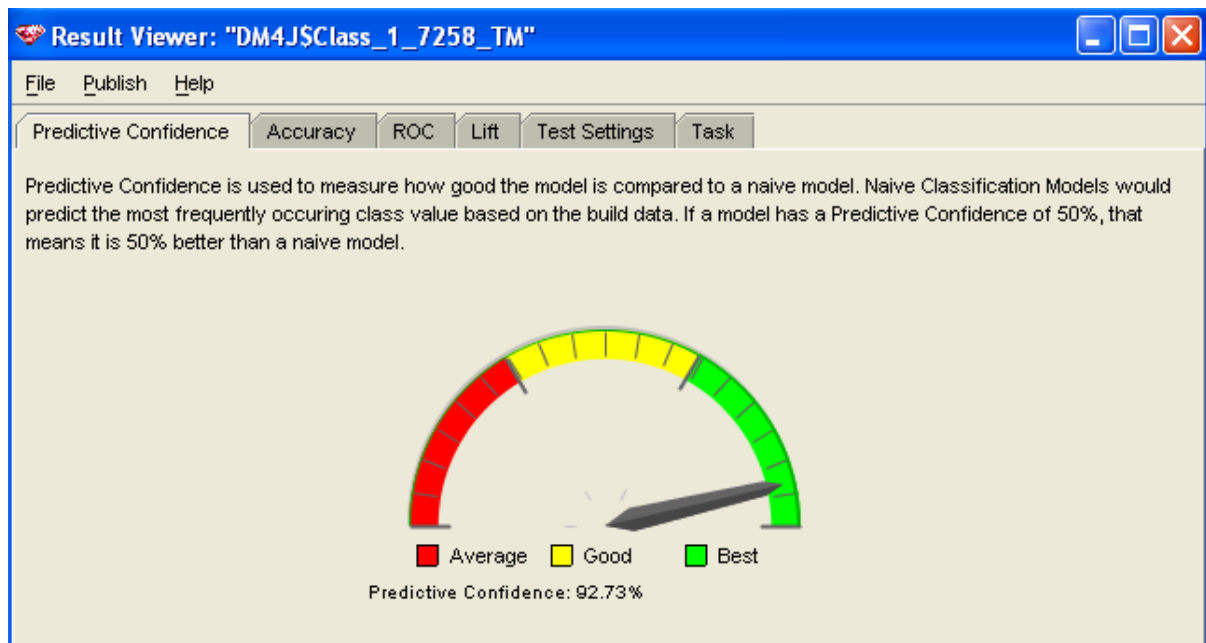


Figure 6.24 predictive confidence of Support Vector Machine algorithm

The accuracy page shown below (figure 6.25) indicates the accuracy of the model when applied to the test data set. The simplest (default) display indicates that from the 30 records with target value of 1, the model correctly predicted 96.67% of them. Likewise, the model correctly predicted 93.1% of the 29 records of target value of 2. The model correctly predicted 98.53% of the 68 records of target value of 3. Perfect predictions (100%) are made for 22 records with target values of 4. The model correctly predicted 82.61% of the 23 records with target values of 5.

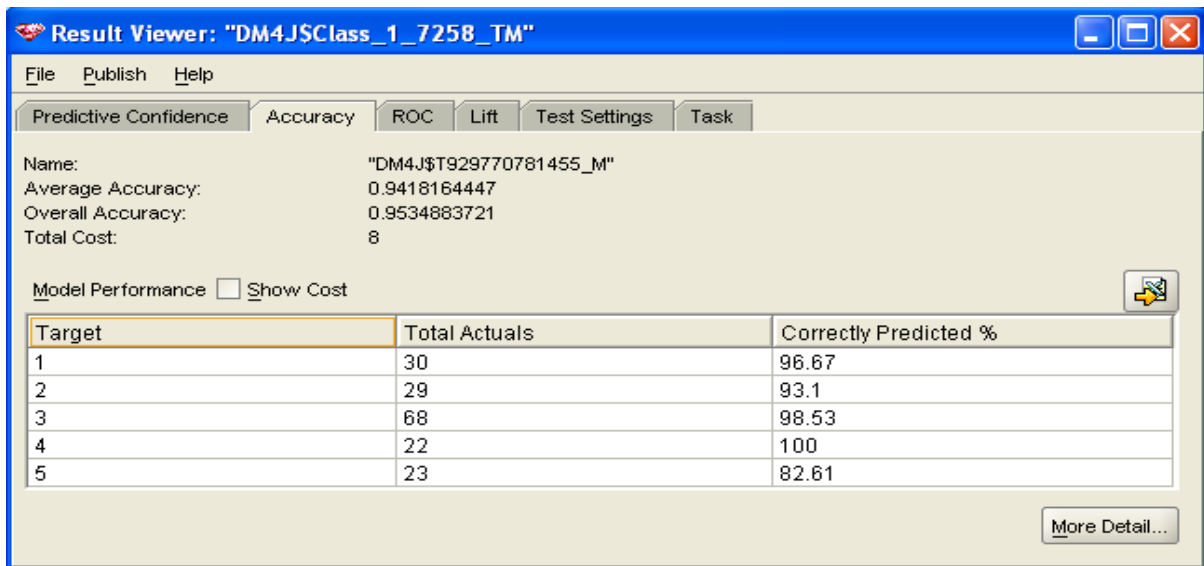


Figure 6.25 Accuracy of Support Vector Machine Algorithm

Cost is another measure in comparing one model to another. It is important to indicate the cost associated with incorrect predictions. The lower the cost is the better the model. Figure 6.26 below shows the cost of the predictions of the Support Vector Machine model.

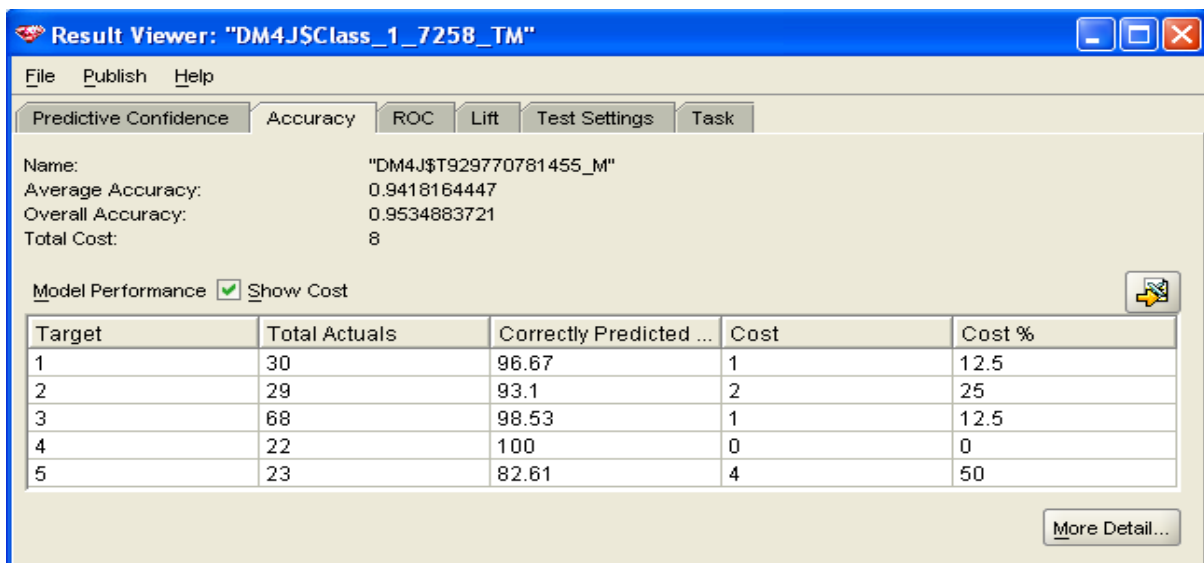


Figure 6.26 Cost of the predictions of Support Vector Machine model

The Confusion Matrix indicates the type of errors that are expected from the model. The values of the *Class* attribute are known and represent the rows of the matrix, and the predictions made by the classification model represent the columns of the matrix. As shown in figure 6.27, the number 1 in row 1/column 2 indicates a false-positive prediction – prediction of 2 when the actual value is 1, while the number 4 in row 5/column 4 indicates a false-negative prediction – prediction of 4 when the actual value is 5.

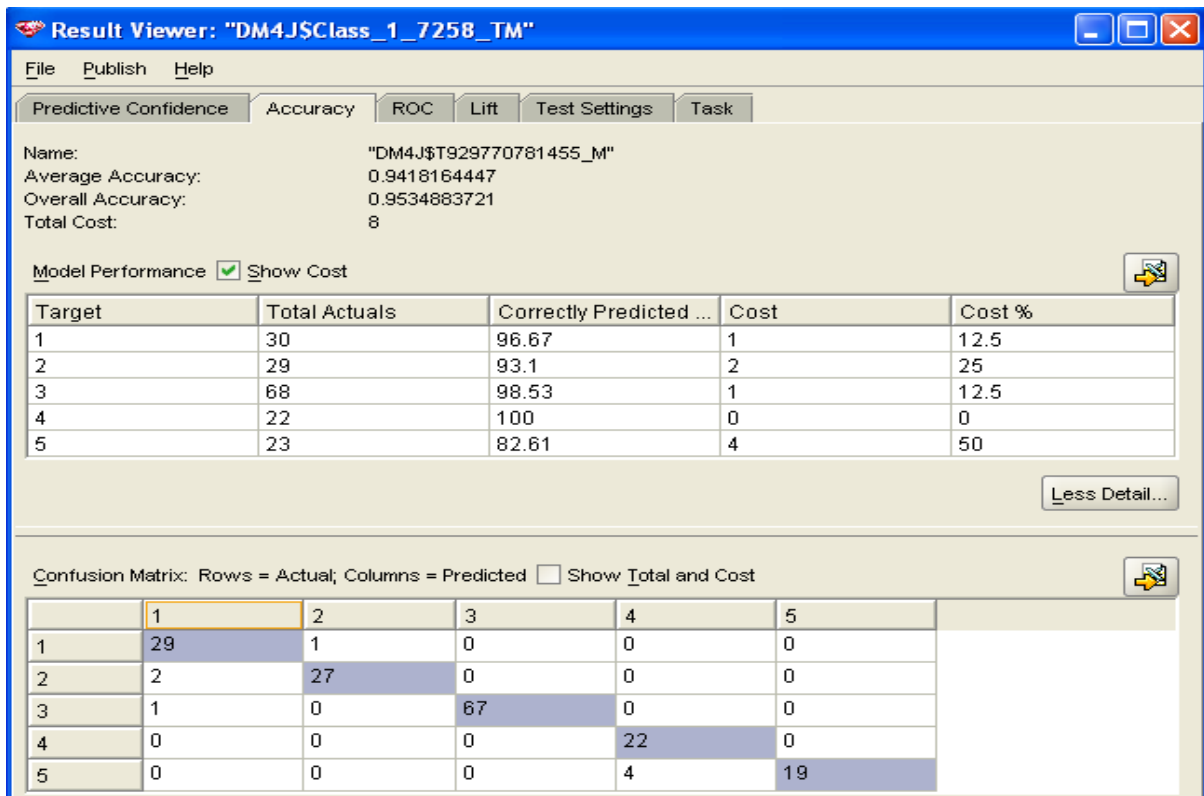


Figure 6.27 the Confusion Matrix of Support Vector Machine model

Table 6.8 below shown an analysis of the outcome of the matrix

Developer Type	Analysis of the model's Confusion Matrix
Excel User	29 of the 30 Excel Users have been correctly predicted. One has been incorrectly predicted as Excel Power User.
Excel Power User	27 of the 29 Excel Power Users have been correctly predicted. Two have been incorrectly predicted as Excel Users.
VBA Developer	67 of the 68 VBA Developers have been correctly predicted. One has been incorrectly predicted as Excel User.
Excel Developer	All Excel Developers have been correctly predicted.
Professional Excel Developer	19 of the 23 Professional Excel Developers have been correctly predicted. Four have been incorrectly predicted as Excel Developers.

Table 6.8 Support Vector Machine Confusion Matrix analysis

The statistics derived from the confusion matrix like the total number of cases in every category, the percentage of correctly predicted cases, and the associated costs are all displayed in figure 6.28.

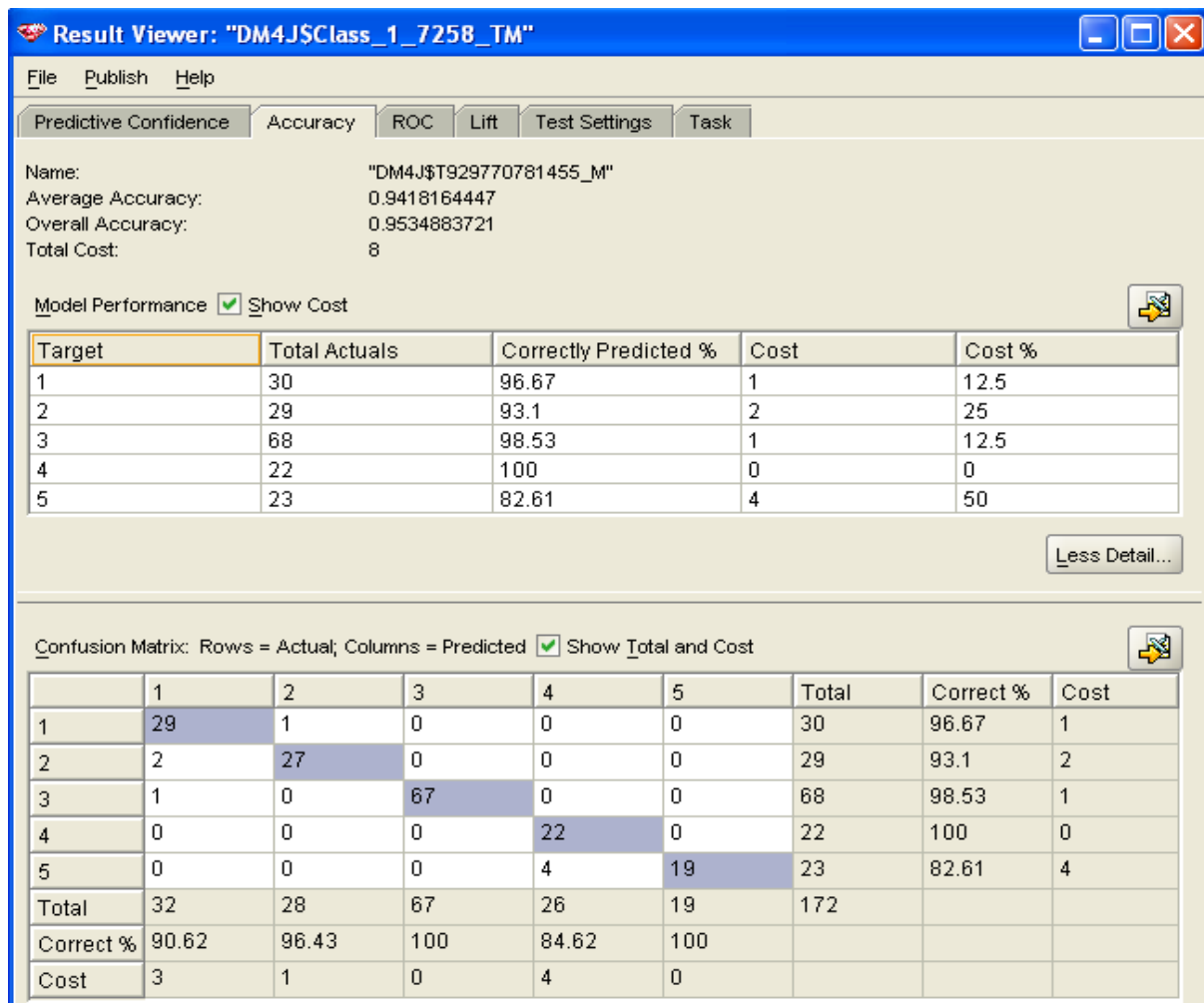


Figure 6.28 the totals and costs of Support Vector Machine Confusion Matrix

6.2.1.1.3.5 Evaluation of the Classification Models' Predictions

The predictions of the four classification models have been compared in terms of predictive confidence, average accuracy, and the average of cost associated with each prediction.

The Adaptive Bayes Network has achieved a prediction confidence of 82.05%, average accuracy of 0.8563768116 and cost of 24.00026. Although both the confidence and accuracy are high, but the average cost is reasonably high compared to the lower costs scored by the Naïve Bayes, Decision Trees, and Support Vector Machine algorithms (see table 6.9). However, it has incorrectly predicted 28 participants. 10 incorrect predictions were displaced by a single rank. 17 predictions were displaced by two ranks, and one was displaced by four ranks. Thus these readings suggest that this model may not be a significant predictor to be applied for the model scoring stage.

The Support Vector Machine has scored better results than Adaptive Bayes Network with confidence and accuracy of 92.73% and 0.9418164447 respectively, and lower cost of 1.6, but with 8 incorrect predictions. 7 of the wrong predictions were displaced by a single rank, and 1 wrong prediction was displaced by two ranks. 6 out of the 8 wrong predictions are shared by the Adaptive Bayes Network model. They also share the same rank displacement. This suggests that it is a reasonably good predictor for the model scoring stage.

The Decision Trees algorithm has achieved higher prediction confidence and average accuracy than both the Adaptive Bayes Network and Support Vector Machine, confidence of 96.36% and accuracy of 0.9708445777, but with lower cost than Adaptive Bayes Network and higher than Support Vector Machine at 4.73114. It has incorrectly predicted 4 participants, which is less than both Adaptive Bayes Network and Support Vector Machine. All of them were displaced by a single rank. The 4 wrong answers predicted by the Decision Trees are shared by the Support Vector Machine model. The same single displacement is also shared by the models. Thus the Decision Trees is a good predictor to be applied for the model scoring stage.

The Naïve Bayes model has achieved the highest prediction confidence of 97.22%, the highest average accuracy of 0.9777411294, and with a reasonable average cost at 3.61008. Although Support Vector Machine has scored slightly better cost at 1.6, but its prediction confidence and accuracy are not as high as the ones Naïve Bayes achieved. Additionally, the

main concern is to select the most accurate model with the highest degree of both confidence and accuracy, preferably with the lowest possible average cost associated. It has only 3 incorrect predictions, which is the lowest number of all models. All of them were displaced by a single rank.

The Naïve Bayes has benefited from applying the Feature Selection method, as the irrelevant or uninformative attributes were eliminated. Thus, the prediction confidence and accuracy of the model are increased.

The algorithm assumes a strong independence between attributes of the dataset. However, based on the author's (expert) good understanding of the data characteristics of the features being considered in the dataset, this has been proved to be valid as the existence and effectiveness of a particular attribute in the dataset does not depend on the existence and effectiveness of any other attributes of a class when building, testing and scoring the model. For example, a user maybe classified as Excel Power User if his/her belonging record in the dataset contains functions like *Date and time functions()*, *Financial functions()*, *Information functions()*, *Logical functions()*, *Maths functions()*, *Statistical functions()*, *Text functions()* and occasional use of VBA macro code constructs. Even if theses attributes depend on the existence of each other or the rest of the attributes in the dataset.

Moreover, although this assumption can be easily violated in reality via having a correlation between attributes in the dataset where these attributes depend on the existence of each other. Naïve Bayes considers all of the attributes of the dataset to individually contribute to the classification of the end users. Thus, this fitted perfectly the type of attributes that constitute the data of interest's dataset, as they are completely independent.

The Naïve Bayes has achieved the best results in terms of predictive confidence, accuracy, cost, and the number of correct predictions. Thus this model is the perfect candidate with the best required characteristics that can provide the best solution to the classification problem. Thus, this model has been selected to undertake the next stage of applying the metadata of the model to the data of interest in order to produce the best possible prediction based classification of excel spreadsheet developers.

Results Algorithm	Prediction Confidence (100%)	Average Accuracy (Correctly predicted)	Average Cost
Adaptive Bayes Network	82.05	0.8563768116	24.00026
Support Vector Machine	92.73	0.9418164447	1.6
Decision Tree	96.36	0.9708445777	4.73114
Naïve Bayes	97.22	0.9777411294	3.61008

Table 6.9 Evaluation of the classification models' predictions

6.2.1.1.4 The Apply Activity

The Naïve Bayes model is determined to be the best solution fits the classification problem. The results are promising as the experiments demonstrated that the NB scored the highest degree of both confidence and accuracy. However, the model is now ready to be applied to the new data (i.e. data of interest) in order to achieve its ultimate goal of the classification. This stage is referred to as *scoring the model*.

A data set named MINING_SOFTWAREUSAGE_APPLY represents the data of interest to be scored by the model. This data set contains the 100 records that were held aside during the data acquisition and preparation stage in order to be used in this final stage of the prediction process. The data set was selected randomly to include 20 students with different skill levels covering the development activities over the period of the six weeks. The selection was made taking into consideration cases that could represent all different types of end users stated in the classification criteria given that the supplied spreadsheets are notionally from different end user types even when actually from the same end user. This is to ensure that the criteria can be fully applied. The selection was made as stated in table 6.10. For example the first 20 cases in which the students were expected to be *Excel Users* contain information that makes these users qualified for this category (see table 6.4). This strategy has been used in the selection of all cases of the data set.

Expectations	Category
20 cases in which the users were expected to be <i>Excel Users</i> .	1
20 cases in which the users were expected to be <i>Excel Power Users</i> .	2
20 cases in which the users were expected to be <i>VBA Developers</i> .	3
20 cases in which the users were expected to be <i>Excel Developers</i> .	4
20 cases in which the users were expected to be <i>Professional Excel Developers</i> .	5

Table 6.10 expectations of the expert for the selected cases of the apply data set.

Although the target values of this data set are unknown, and the algorithm is required to achieve this task. An expert can manually assign expected target values for each case (record). However, the author of the project has played this role via analysing the data records. The expectations have been made based on the author's good understanding of the data characteristics, experience gained throughout the phases of the project, and the relatively small size of the data set. These expected target values will not take part in the model applying phase. The results (i.e. outcome generated from the model applying phase) can then be compared against these values in order for the author to check the validity of the results.

Considering the facts that the expert is fully aware of the attributes that contain useful information within the data set and the independence of the attributes one of another for each target value (class), the Naïve Bayes independence assumptions feature has been taken into consideration when making the selection. That is in order to prevent any invalidation problems. However, the Naïve Bayes was designed to work with data sets in which the attributes are independent of one another for each target value, but it is also agreed that the algorithm is surprisingly effective in practice even when the independence assumption is not valid. Moreover, the Naïve Bayes considers all of the attributes of the data set to independently contribute to the classification of the end users.

The apply activity is based on the build activity that was used to create the model. Thus, all the information required about data preparation and model metadata will be passed to the apply activity.

When the model is applied to the data of interest, a probability value is generated for each predicted target value; this indicates the confidence of the prediction made. The cost

associated with every individual prediction is also generated, lower cost indicates higher probability, and it represents the cost of any incorrect prediction made by the model. A ranking value is assigned to each prediction. This value in this case is always 1 for all cases, as the model is required to predict the most probable target value for each case.

The format shown in table 6.11 illustrates the results (most probable predictions) made by Naïve Bayes model when applied to the data of interest. The table contains on each row, the identifier, the predicted value, the probability (confidence of the prediction), the cost of the prediction, the rank given to each prediction, and the *Author_ID*, this is added to represent the owner of the case. The results include both the correct and incorrect predictions. For example, the candidate 218 (highlighted in blue) was correctly predicted to be a VBA Developer with probability of 1 (100% prediction confidence), with the lowest cost of 0, and ranking value of 1 indicating that this is the most probable target value for this developer.

The candidate 437 (highlighted in yellow) was incorrectly predicted as *Professional Excel Developer* (prediction of 5) with a low probability of 0.5962, high cost of 3.5107, and with a single rank displacement. Whereas the analysis of the data record for this candidate clearly indicates that *Excel Developer* is the correct prediction (prediction of 4). This shows the first of two cases where the algorithm generated wrong predictions. The candidate 512 (highlighted in orange) was incorrectly predicted as *Excel Developer* (prediction of 4) with a probability of 0.982, a reasonably high cost of 0.1258, and with a single rank displacement. In spite the probability of the prediction is high; the analysis of the data record for this candidate shows that the correct prediction is *Professional Excel Developer* (prediction of 5). Thus, these are the two cases in which the algorithm generated incorrect predictions, as the average accuracy of the model is 0.9777411294/1 (i.e., it is not 100% accurate). The table below shows only part of the results, the complete set can be found in appendix F. Table 6.12 shows the analysis of the apply activity results.

Apply Output Table:					
Fetch Size: 100		Refresh			
DMR\$CASE_ID	PREDICTION	PROBABILITY	COST	RANK	AUTHOR_ID
409,002,816	3	1	0	1	216
409,003,072	3	1	0	1	217
409,003,136	3	1	0	1	218
409,003,136	3	1	0	1	219
409,003,520	3	1	0	1	220
509,049,280	4	0.9998	0.0011	1	426
509,049,568	4	0.982	0.1258	1	427
509,052,864	4	0.982	0.1258	1	428
509,053,600	4	0.9807	0.1346	1	429
509,055,520	4	0.982	0.1258	1	430
509,058,336	4	0.9807	0.1346	1	431
509,059,200	4	0.9998	0.0011	1	432
509,116,480	4	1	0	1	433
509,116,736	4	0.9607	0.2742	1	434
509,117,696	4	0.9732	0.187	1	435
509,117,984	4	0.982	0.1258	1	436
509,120,800	5	0.5962	3.5107	1	437
509,126,304	4	0.9607	0.2742	1	438
509,127,392	4	0.982	0.1258	1	439
509,133,632	4	0.982	0.1258	1	440
509,136,128	4	0.982	0.1258	1	441
509,160,928	4	0.9384	0.4304	1	442
509,161,024	4	0.982	0.1258	1	443
509,161,088	4	0.982	0.1258	1	444
509,165,184	4	0.982	0.1258	1	445
609,049,600	5	1	0	1	507
609,053,632	5	0.9997	0.003	1	508
609,055,552	5	0.999	0.0089	1	509
609,058,368	5	1	0	1	510
609,058,368	5	0.9999	0.0005	1	511
609,059,200	4	0.982	0.1258	1	512
609,060,480	5	1	0	1	513
609,110,528	5	0.9999	0.0011	1	514
609,116,480	5	0.9943	0.0496	1	515

Table 6.11 The Naïve Bayes apply activity results

Prediction	Analysis of the apply activity results	Developer Type
1	20 cases have been predicted as Excel Users with prediction confidence of 1 (100%), cost of 0, and ranking value of 1.	Excel User
2	20 cases have been predicted as Excel Power Users with prediction confidence of 1 (100%), cost of 0, and ranking value of 1.	Excel Power User
3	20 records have been predicted as VBA Developers with prediction confidence of 1 (100%), cost of 0, and ranking value of 1.	VBA Developer

4	20 cases have been predicted as Excel Developers. One of them is with prediction confidence of 1 and cost of zero. 19 with level of confidence higher than 0.93, and various low levels of costs. One case (highlighted in Orange) was predicted by the NB as Excel Developer in a place where it was expected by the author to be Professional Excel Developer. The 20 cases have been assigned a ranking value of 1.	Excel Developer
5	20 cases have been predicted as Professional Excel Developers. 10 of them are predicted with predictive confidence of 1 and at the cost of zero. 9 cases with level of confidence higher than 0.95 and various low levels of cost. One case (highlighted in Yellow) has been predicted with low predictive confidence of 0.5962 and fairly high cost of 3.5107, this case was predicted by the NB to be Professional Excel Developer in a place where it was expected by the author to be Excel Developer. The 20 cases have achieved a ranking value of 1.	Professional Excel Developer

Table 6.12 the analysis of Naïve Bayes apply activity results

From the analysis of the results above, Naïve Bayes has performed significantly well classifying the Excel developers, and proved a highly effective and accurate classifier on real data applications. Thus, this algorithm can be relied on to be robust when used for Excel users' classification purposes.

Chapter 7 - Conclusion and Future Work

This chapter concludes the thesis with a summary of the objectives that have been considered and how these objectives have been met in order to achieve the goal of the development of MACS. The contributions of this work to knowledge and future work are discussed.

The thesis presents a novel approach to develop a distributed Multi-Agent based data gathering and classification system that automatically monitors excel spreadsheet applications by content and classify the developers according to their knowledge of excel and VBA macro code constructs. However, the research question addressed in the research is:

Is it feasible to automatically and autonomously monitor spreadsheet developers over a network based on the utilization of the software Multi-Agent technology in such a way that makes management capable to accurately classify spreadsheet developers and allow for precise tailor training activities for future spreadsheet development.

In this new approach, the thesis has presented the research results obtained from focusing on the task to develop MACS that gathers data from multiple spreadsheet developers over a network, and the data is then analysed in order to produce accurate predictions of excel spreadsheet developers. It has been demonstrated that the .NET Windows Service based agents can be utilized together with the Microsoft's FileSystemWatcher component to function automatically and autonomously to provide continuous and periodic monitoring of Excel spreadsheet development activities. The Prometheus agent oriented methodology and its accompanying Prometheus Design Tool (PDT) have been utilized efficiently for the design of the software agents; the PDT provides an instrument for cross checking the artefacts to ensure a consistent agent specification.

It has also been demonstrated that the .NET WCF services technology can be utilized successfully for the distribution of the agents over the WWW in order to satisfy the monitoring, data gathering, and classification of the multiple developer aspect. This technology provided the system with extra advantages such as the asynchronous calls between the client and server agents residing in different machines, and the provision of reliable end-to-end streaming transfer between SOAP endpoints. Resulting in, the Multi-Agent Classification System (MACS) is developed. MACS has been built based on SOA, this supplied the system with more flexibility and the interaction between the agents is simplified.

The work has demonstrated results gained from the experiments focused on Oracle data mining algorithms performed in oracle 10g environment. Naive Bayes classifier has proved success in providing accurate predictions of excel spreadsheet developers.

The MACS seeks to contribute to the fields of knowledge: End User Computing, Multi-Agent Systems, Distributed Programming, Spreadsheet Development, and Data Mining by providing an automatic and accurate classification tool to managers and those end users that develop applications in the spreadsheet domain.

The development of MACS has proved successful and provided evidence that the concepts of the research are valid and has provided a significant enhancement to the literature available to the research and development communities. Successful evaluation of MACS has also been demonstrated through the use of real-time data collected from Excel spreadsheet developers.

7.1 How the objectives were met

All the objectives of the research defined in the introduction chapter were accomplished and the results were very satisfactory.

1. To consider the role that software multi-agents play within the context of a user classification multi-agent.

A comprehensive literature review has been undertaken to cover the separate research topic areas of end user computing and software agents with emphasis on .NET Windows Service based agents and investigate how these can be together utilized to promote more effective end-user computing. Another investigation has been undertaken of research material relating to the concept of classification of end-users, in particular the spreadsheet application developers. The findings of the process were discussed in chapter two of the thesis. In particular, a lack of automatic and accurate classification tool is highlighted as prominent weakness in the spreadsheet development domain. The background and related work research showed that an effective solution is the development of a Multi-Agent based classification system to overcome this problem. To automate the classification, the process required a toolset to monitor and collect data from spreadsheet developer machines, a toolset to transform and filter the data ready for classification, and a data mining application to produce the appropriate category for each spreadsheet developer.

2. To identify the agents needed to comprise the multi-agent system required to develop a classification agency.

Two agents were primarily identified namely the Monitoring Agent (MA) and the Database Updater Agent (DUA).

- The MA was used to automatically monitor and gather data from Excel spreadsheet applications.
- The DUA was used to collect the data gathered by the Monitoring Agent, filters the data, and then connects to Oracle server database for data transfer to the appropriate tables.

3. To design the required agents using an appropriate methodology.

Chapter four shows how Prometheus agent oriented methodology was utilized efficiently for the design of both the MA and DUA agents. It was chosen because it is intended to be a practical methodology. As such it aims to be complete providing everything needed to specify and design the agents. Additionally, Prometheus is detailed and facilitates tool support, this tool support exists in the form of Prometheus Design Tool (PDT); PDT was developed to support the design of Multi-Agent systems using agent oriented approach.

4. To develop a prototype multi-agent using an appropriate programming language.

The agency was created in Microsoft's Visual Studio .NET 2008 using Visual C# as the programming language. The prototype was divided into the server-side and the client-side software applications as illustrated in Chapter 3. The MAs agents that made up the server-side application are concerned with collecting and storing data temporarily into local data stores. Microsoft's FileSystemWatcher component was utilized to add the detection capability to the agents, as these agents are implemented to continuously listen to any excel spreadsheets can be accessed by developers during their life time. The DUA agent that represents part of the client-side was implemented to run by the User Agent when needed. The gathered data is then collected by this agent, filtered and transferred to Oracle server database to be stored permanently for further processing that leads to the classification of the Excel developers.

5. To identify an appropriate end user working environment in order to test the classification system.

Microsoft Excel application was used as workbench for classification in this research project due to fact that many companies rely on spreadsheets as a key tool in their financial reporting and operational processes, and because it offers functionality and variety of features that could be used to express the different levels of user's development. [16] States that the allocation is dependent on their experience and knowledge of both Excel and Excel macro language Visual Basic for Application (VBA). The categories are explained in table 2.6. Chapter six illustrated how Oracle Data Mining's Naive Bayes algorithm managed to successfully achieve the categorization task.

6. To utilize the system to automatically monitor end users development activities by content.

The MAs were utilized successfully to monitor and read excel spreadsheets by content in terms of file and author properties (File Usage), all functions and formulas a spreadsheet contains, and VBA macro code constructs (Object and Code Usage Patterns). Table 4.1 and appendix C show the features of the file properties and object and code usage patterns the MAs are interested in and required to monitor.

7. To autonomously run the Multi-Agent system locally as a .NET Windows Services process.

The MAs were implemented to start automatically as a .NET windows service process when users log on. Thus they have the capability to run autonomously in the background as a service running in its own thread space, and gather data without user intervention as long as the operating system is running.

8. To distribute the agents over the World Wide Web using .NET Windows Communication Foundation (WCF) services technology.

.NET WCF services were used for the distribution of the MAs agents in order to satisfy the monitoring and classification of the multiple developer aspect. The MAs were distributed to collect data from multiple resources over the network of the University of Glamorgan. The

WCF technology was also utilized to provide the communication abilities over TCP protocol between the MAs and the DUA agents of the system.

9. To identify an efficient communication method for data exchange between agents.

MACS was developed with respect to SOA architecture. Therefore, the SF is considered as a mandatory and vital part of the system. It is being used to advertise the WCF services the MAs expose. In this case, the MAs do not take part in the communication process. The DUA accesses the MAs through the WCF services they expose using the WCF client proxy, regardless of the locations in which the MAs are residing on the server-side application of the system. All the DUA needs to know is the URL addresses of the WCF services the MAs expose in order to make successful calls. Thus, this method added efficiency to the communication process between the agents.

10. To utilize Oracle data mining classification algorithms to achieve the classification phase of the system.

The proposed MACS has been implemented to provide an automatic and accurate classification of end users developing applications in the spreadsheet domain (chapters five and six). The Oracle data mining classification models: Naive Bayes, Adaptive Bayes Networks, Decision Trees, and Support Vector machine were all tested to distinguish between the five different developer types illustrated in table 6.4. The predictions of the four models have been compared in terms of the predictive confidence, average accuracy, and average cost associated with the prediction. The results of the comparison showed that the Naive Bayes model achieved the best results with prediction confidence of 97.22%, average accuracy of 0.978 of 1, and reasonably low cost of 3.61 (Table 6.9). This demonstrates that it is possible to implement the automated solution with a high degree of accuracy.

11. To determine criteria for the effective evaluation of the Multi-Agent classification system.

The MACS has proved successful, reliable and valid. The amount of 426 spreadsheets developed by 80 students at the University of Glamorgan was used for building and evaluating the models. 100 spreadsheets that represent the data of interest were used for the

applying activity process using the most accurate model (Naive Bayes). The analysis of the apply activity results (see table 6.11 for partial results or appendix F for complete results) showed that Naive Bayes has proved a high degree classifier on real data applications. Thus, MACS utilizing Naive Bayes classifier can be relied on to be robust and accurate when used for excel users' classification purposes.

7.2 Contribution to Knowledge

The research resulted in three contributions to knowledge in the research areas of End User Computing, Multi-Agent Systems, Distributed Programming, Spreadsheet Development, and Data Mining.

1. The development of a Software Multi-Agent based system that has the capability to automatically and autonomously monitor Microsoft Excel spreadsheets by content.

The utilization of .NET Windows Service based software agents together with the Prometheus agent oriented methodology has demonstrated that the agent technology can be successfully used to automatically and autonomously monitor developers of Microsoft excel spreadsheets by content. This contribution was presented in conference paper that addressed the development of a Windows Service agent to analyse spreadsheet macro and function complexity [57]. The contribution in this area was carried out further with an IEEE conference paper discussed the design and implementation of a .NET Windows Service based agents that function autonomously and automatically collect data for subsequent analysis and graphical presentation of spreadsheet data [93].

2. The Utilization of Microsoft's .NET WCF services technology together with SOA architecture for the distribution of the agents over a network.

.NET WCF services technology together with SOA concept have proven to be a success for this project. They have been used to build a distributed system in which the agents can effectively communicate with each other over the network of the University of Glamorgan in order to achieve the monitoring and data collection tasks. Resulting in, a Mini-Network was established in order to satisfy the monitoring, data collection, and classification of the multiple developer aspect.

3. The utilization of Oracle data mining algorithms to automatically and accurately achieve the classification of Excel spreadsheet developers.

The results obtained from the data mining experiments focused on analysing the contents of excel spreadsheets clearly demonstrated that: it is possible to use Oracle data mining classification algorithms, in particular Naïve Bayes algorithm to provide highly accurate and reliable predictions of excel spreadsheet developer categories.

7.3 Future Work

Although the aim and objectives have been successfully accomplished, but the work leading to this thesis has generated some interesting and promising ideas, these ideas are worth exploring further as future work.

- At present, work on the implementation has concentrated on the development of a distributed Multi-Agent based monitoring tool that automatically and autonomously gather data from multiple spreadsheet developers over network. Excel spreadsheets have been used as a test bench environment for user data gathering. A future environment can be to allow the production of a user profile that includes the use of other Microsoft Office applications such as Word, Access, PowerPoint, etc. Applications other than Microsoft Office can also benefit from the system such as PDF applications.
- The tool can be improved by including additional properties to the agents; such as being reactive, and social; this will be useful in that it will make the agents more intelligent. Furthermore, the interaction between agents can be improved in terms of 'inform', 'request', and 'agree' and in terms of human interaction types such as negotiation and coordination.
- Since the design of MACS is based on SOA architecture, a good extension is to implement the SF as a WCF service, thus the DUA agent can use common and more efficient technique to find and invoke the methods that the MAs agents expose.
- Currently, the WCF services have been used for the distribution of the agents on computers running Microsoft Windows operating system. A future direction is to apply the interoperability aspect to the system. The WCF native messaging protocol SOAP, which is an open standard, provides the opportunity for the WCF services to

interact with different technologies running on different platforms and non-Windows operating systems [119].

- Another possible future direction for the project is the security of spreadsheets. The MAs agents at present are configured to run automatically and continuously in the backgrounds to monitor spreadsheet's file properties such as author name, document name, date created, date last saved, and last saved by, together with the type and number of functions used and type and number of VBA macro code constructs used. Data collected from every single monitoring session for every spreadsheet is stored as a single record in oracle database. This data can then be retrieved and analysed to check the changes have been made and the author(s) responsible for the changes. This development provides the spreadsheet assessor with a complete history of all changes and when the changes have occurred during the lifetime of the spreadsheet.
- The MACS can be applied to track and categorize the learning of users developing spreadsheets on a European Computer Driving Licence (ECDL) course. Some work has been undertaken to assess the complexity of the spreadsheets for students studying the ECDL at different levels [92]. The applying process requires changes to the classification criteria MACS used in order to reflect the three levels of ECDL accomplishment. This also involves some changes on the data mining process, as Oracle data mining algorithms will need to be trained based on the ECDL criteria. The adjusted MACS will then offer course administrators a quick identification of the levels of development activities for every candidate.

References

- [1]. .NET Framework Conceptual Overview, .NET Framework Developer Centre, MSDN, <http://msdn.microsoft.com/en-us/netframework/default.aspx>, (Last visited, 2010).
- [2]. AGENTCITIES - a global, collaborative effort to construct an open network of on-line systems hosting diverse agent based services, <http://www.agencycities.org>, (Last visited 2010).
- [3]. AHMED, S. (2007) Analysis of Workplace Surveillance in a Quest for an Ethical Stance. *Journal of Business Systems, Governance and Ethics*, 2,4, 15-26.
- [4]. Aignesberger Software GMBH, Monitoring PDF/Word/Excel files, <http://www.aignes.info/blog/?p=292>, (Last visited, 2010).
- [5]. AMOROSO, D. & CHENEY, P. H. (1992) Quality End User-Developed Applications: Some Essential Ingredients. *Data Base*, 1-11.
- [6]. AULT, M., LIU, D. & TUMMA, M. (2003) *Oracle Database 10g New Features: Oracle 10g Reference for Advanced Tuning and Administration*, kittrell, north Carolina, USA, Rampant TechPress.
- [7]. AWAD, M. A. (2005) A Comparison between Agile and Traditional Software Development Methodologies. *School of Computer Science and software Engineering*. The University of Western Australia.
- [8]. BELLIFEMINE, F., POGGI, A. & RIMASSA, G. (1999) JADE - A FIPA-compliant agent framework. *Proceedings of the 4th International Conference on the Practical Applications of Agents and MultiAgent Systems PAAM99*. The Practical Application Company Ltd., Pages: 97-108.
- [9]. BLACKWELL, A. (2002) What is Programming? *14th Workshop of the Psychology of Programming Interest Group*. Brunel University.
- [10]. BOEHM, W., ABTS, C., BROWN, W., CHULANI, S., BRADFORD, K., HOROWITZ, E., MADACHY, R., DONALD, J. & STEECE, B. (2000) Software Cost Estimation with COCOMO II. *Prentice-Hall*.
- [11]. BOND, A. H. & GASSER, L. (1988) *Readings in Distributed Artificial Intelligence*, San Mateo, Morgan Kaufmann Publishers.
- [12]. BOOCH, G., JACOBSON, I. & RUMBAUGH, J. (1998) Unified Modeling Language 1.3, White paper. Rational Software Corp.
- [13]. BRADSHAW, J. M. (1997) *Software Agents*, Cambridge, MIT Press.

- [14]. BRADSHOW, J. M. (1997) An introduction to software agents. IN BRADSHOW, J. M. (Ed.) *Software agents*. Cambridge, MIT Press.
- [15]. BRIGHAM, F. & DAVES, R. (2009) *Intermediate Financial Management*, USA, South-Western, Cengage Learn
- [16]. BULLEN, S., BOVEY, R. & GREEN, J. (2009) *Professional Excel Development: The Definitive Guide to Developing Applications Using Microsoft Excel and VBA*, Upper Saddle River, Addison-Wesley.
- [17]. BUSSINK, D. (2004) A Comparison of Language Evolution and Communication Protocols in Multi-agent Systems. *1st Twente Student Conference on IT, Track C - Intelligent Interaction*.
- [18]. CAMELEON (Communication Agents for Mobility Enhancements in a Logical Environment of Open Networks), ACTS, Project AC314, <http://www.comnets.rwth-aachen.de/>.
- [19]. CAMMARATA, S., MCARTHUR, D. & STEEB, R. (1983) Strategies of Cooperation in Distributed Problem Solving. *Proceedings of the Eight International Joint Conference on Artificial Intelligence*, pages 767-770.
- [20]. CARDELLINI, V., CASALICCHIO, E. & COLAJANNI, M. (January 03 - 06, 2001) A Performance Study of Distributed Architectures for the Quality of Web Services. *34th Annual Hawaii international Conference on System Sciences (Hicss-34)*. Washington, IEEE Computer Society.
- [21]. CHIRA, C. (2003) Software agents. (IDIMS Report).
- [22]. CHRIS, P., DENNIS, M., SHAWN, C. & AMIT, B. (2007) *Pro WCF: Practical Microsoft SOA Implementation*, USA, APress.
- [23]. Codasyl end-user facilities committee status report. Information Management Two, North Holland. 1979, 137-163.
- [24]. Computer Science and Telecommunications Board. Being Fluent with Information Technology, National Research Council, 1999.
- [25]. COTTERMAN, W. & KUMMAR, K. (1989) User Cube: A Taxonomy of End Users. *Communication of the ACM*, 32, 1313-1320.
- [26]. CUNNINGHAM, W. Manifesto for Agile Software Development, <http://www.agilemanifesto.org/>, Accessed on 17/02/2011.
- [27]. DAUGHERTY, C. (1999) Employers use high-tech tools to watch employees. *New Orleans City Business*, 19(27), 10-12.
- [28]. Dictionary of Computing. Oxford University Press, 1983.

- [29]. Document Content Description (DCD) for XML, <http://www.w3.org/TR/NOTE-dcd>, (Last visited 2009).
- [30]. DOLL, W. & TORKZADEH, G. (1988) The Measurement of end-User Computing Satisfaction. *MIS Quarterly*, 12(2), 259-274.
- [31]. EDWARDS, J., PEOPLESOFT & SIEBEL (2006) Oracle 10g Release 2 Data Mining Tutorial.
- [32]. ETZIONI, O. & WELD, D. S. (1995) Intelligent agents on the Internet: Fact, fiction and forecast. *IEEE Expert*, 10(4), 44-49.
- [33]. FACTS (FIPA Agent Communication Technologies and Services), ACTS Project AC317., <http://cordis.europa.eu/infowin/acts/rus/projects/ac317.htm>.
- [34]. FAUSTO, G., JOHN, M. & PERINI, A. (2002) The Tropos software development methodology: Processes, models and diagrams. *In Third International Workshop on Agent-Oriented Software Engineering*.
- [35]. FININ, T., LABROU, Y. & MAYFIELD, J. (1997) KQML as an agent communication Language. IN GRADSHOW (Ed.) *Software Agents*. Cambridge, MIT Press.
- [36]. FIPA (2002) FIPA Abstract Architecture Specification.
- [37]. FIPA (2004) FIPA Agent Management Specification.
- [38]. FIPA (Foundation of Intelligent Physical Agents), <http://www.fipa.org>, (last visited 2009).
- [39]. FIPA Agent Message Transport Service Specification, <http://www.fipa.org/specs/fipa00067/>, (last visited 2010).
- [40]. FOWLER, M. The New Methodology, <http://www.martinfowler.com/articles/newMethodology.html>, Accessed on 17/02/2011.
- [41]. GENESERETH, M. & KETCHPEL, S. (1994) Software agents. *Communications of the ACM*, 37(7), 48-53.
- [42]. GOODWIN, R. (1993) Formalizing properties of agents. *L. Logic Comp.* Pittsburgh, PA: Carnegie-Mellon University, School of Computer Science., 5(6), 763-781.
- [43]. GOVAERT, G. (2009) *Data Analysis*, ISTE Ltd and John Wiley & Sons, Inc.
- [44]. GOVINDARAJULU, C. (2003) End Users: Who are They? *Communication of the ACM*, 46, 152-159.

- [45]. GRADY, B., JACOBSON, I. & RUMBAUGH, J. (1998) Unified Modeling Language. *White paper, Rational Software Corp.*
- [46]. GREEN, S., HURST, L., NANGLE, B., CUNNINGHAM, P., SOMERS, F. & EVANS, R. (1997) Software Agents: A Review. Trinity College Dublin.
- [47]. GUPTA, S. (2007) A Performance Comparison of Windows Communication Foundation (WCF) with Existing Distributed Communication Technologies. <http://msdn.microsoft.com/en-us/library/bb310550.aspx>. Microsoft Corporation - MSDN, (Last visited, 2010).
- [48]. HALSTEAD, M. H. (1977) *Elements of Software Science*, New York, Elsevier North-Holland.
- [49]. HAN, J. & KAMBER, M. (2001) *Data Mining: Concepts and Techniques*, San Francisco, Morgan Kaufmann Publishers.
- [50]. HAN, J. & KAMBER, M. (2006) *Data Mining: Concepts and Techniques*, San Francisco, Morgan Kaufmann Publishers.
- [51]. HASAN, J. & DURAN, M. (2006) *Expert service-oriented architecture in C# 2005*, Apress.
- [52]. HENNING, M. (August 2008) The Rise and Fall of Corba. *Commun. ACM*, 51, 52-57, URL: <http://dx.doi.org/10.1145/1378704.1378718>.
- [53]. HOC, M. & NGUYEN-XUAN, A. (1990) Language Semantics, Mental Models and Analogy. *Psychology of Programming. Academic Press*, 139–156.
- [54]. HOLE, S. & MCPHEE, D. (2005) Building .NET Software Agent to Facilitate Automatic collection of End-User Data. *2nd International Conference on Computer Science and Information Systems*. Athens, Greece.
- [55]. HOLE, S. & MCPHEE, D. (2006a) Building a .NET Agent Software Agent to Facilitate Automatic collection of End-User Data. *2nd International Conference on Computer Science and Information Systems*. Athens, Greece, Athens Institute for Education and Research.
- [56]. HOLE, S. & MCPHEE, D. (2006b) Constructing a Windows Service Based Agent to Facilitate the Automatic Collection of End User Desk-Top Data. *3rd International Conference on Telecommunications and Computer Networks*. Portsmouth, United Kingdom.
- [57]. HOLE, S., MCPHEE, D. & MHEREEG, M. (2008) Developing a Windows Service Agent to Analyze Spreadsheet Macro and Function Complexity. *4th*

- International Conference on Computer Science and Information Systems*. Athens, Greece, Athens Institute for Education and Research.
- [58]. HORN, E., KUPRIES, M., REINKE, T. & IN (1999) Properties and models of software agents and prefabrication for agent application systems. *Proceedings of the International Conference on System Sciences (HICSS-32), Software Technology Track*.
 - [59]. HOWARD, C. (1998) Security banquet. *Canadian Business*, 71(13), 80.
 - [60]. IBRAHIM, D. & MISIC, V. B. (2006) Service Views: a Coherent View Model of the SOA in the Enterprise. *IEEE International Conference on Services Computing (SCC'06)*.
 - [61]. Introduction to Windows Service Applications, MSDN, [http://msdn.microsoft.com/en-us/library/d56de412\(VS.80\).aspx](http://msdn.microsoft.com/en-us/library/d56de412(VS.80).aspx), (last visited, 2010).
 - [62]. JACOBSON, I., BOOCH, G. & RUMBAUGH, J. (1999) *Unified Software Development Process*, Addison-Wesley.
 - [63]. JAMES, M., SAMEER, T., MICHAEL, S. & SUNIL, M. (2003) *Java Web Services Architecture*, San Francisco, CA: Morgan Kaufmann Publishers.
 - [64]. JANCA P, C. (1995) Pragmatic Application of Information Agents: BIS Strategic Decisions.
 - [65]. JELEN, B. & SYRSTAD, T. (2004) *VBA and Macros for Microsoft Excel*, Indianapolis, Indiana, Que.
 - [66]. JENNINGS, N. & WOOLDRIDGE, M. (1996) Software agents. *IEE Review*, 17-20.
 - [67]. JENNINGS, N. R. & WOOLDRIDGE, M. (1998) *Agent Technology: Foundations, Applications, and Markets*, Berlin, Springer.
 - [68]. JENNINGS, N. R., NORMAN, T. J. & FARATIN, P. (1998) ADEPT: An agent-based approach to business process management. *ACM SIGMOD Record*, 27(4), 32-39.
 - [69]. JENNINGS, N. R., SYCARA, K. & WOOLDRIDGE, M. (1998) A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems Journal*, 1(1), 7-38.
 - [70]. JURIC, M. B., ROZMAN, I., BRUMEN, B., COLNARIC, M. & HERICKO, M. (2006) Comparison of performance of web services, WS-security, RMI, and RMI-SSL. *Journal of Systems and Software*, 79(5), 689-700.

- [71]. KENG, S. & HALPIN, T. A. (2001) *Unified modeling language: systems analysis, design and development issues*, USA, Idea Group Publishing.
- [72]. KHANH, H. D. & MICHAEL, W. (2003) Comparing agent-oriented methodologies. IN PAOLO, G. & MICHAEL, W. (Eds.) *Proceedings of the Fifth International Bi-Conference Workshop on Agent-Oriented Information Systems*. Melbourne, Australia.
- [73]. KHANH, H. D. (2003) Evaluating agent-oriented software engineering methodologies, Master's thesis, *School of Computer Science and Information Technology*. Melbourne, Australia, RMIT University, (supervisors: Michael Winikoff and Lin Padgham).
- [74]. KQML (Knowledge Query Meta Language), <http://www.cs.umbc.edu/kqml/>, (last visited 2009).
- [75]. KRAFZIG, D., BANKE, K. & SLAMA, D. (2005) *Enterprise SOA: Service-oriented Architecture Best Practices*, Englewood Cliffs, Prentice Hall PTR.
- [76]. KRUCHTEN, P. (1999) *Rational Unified Process-An Introduction*, Addison-Wesley.
- [77]. LESSER, V. R., DURFEE, E. H. & CORKILL, D. (1989) Trends in Cooperative Distributed Problem Solving. *IEEE Transactions on Knowledge and Data Engineering*.
- [78]. LIN, P. & MICHAEL, W. (2002) Prometheus: A methodology for developing intelligent agents. *Third International Workshop on Agent-Oriented Software Engineering*.
- [79]. LIN, P. & MICHAEL, W. (2002) Prometheus: A pragmatic methodology for engineering intelligent agents. *In Proceedings of the OOPSLA 2002 Workshop on Agent-Oriented Methodologies*.
- [80]. LIN, P. & MICHAEL, W. (2004) The Prometheus Methodology.
- [81]. List of Worksheet Functions (By Category), <http://office.microsoft.com/en-us/excel-help/list-of-worksheet-functions-by-category-HP010079186.aspx>, Microsoft Corporation, (last visited, 2010).
- [82]. LUCK, M., ASHRI, R. & D'INVERNO, M. (2004) *Agent-Based Software Development*, London, Artech House.
- [83]. MAES, P. (1994) Agents that reduce work and information overload. *Communications of the ACM*, 37(7), 30-40.

- [84]. MAHMOOD, A. (2003) *Advanced Topics in End User Computing Idea Group Inc (IGI), USA.*
- [85]. MARGARET, T., RAMKUMAR, K., MARK, H., DENIS, M., GEORGE, T., SHIBY, T. & PETER, S. (2005) Oracle Data Mining Concepts, 10g Release 2 (10.2).
- [86]. MARGARET, T., RAMKUMAR, K., MARK, H., DENIS, M., GEORGE, T., SHIBY, T. & PETER, S. (2008) Oracle Data Mining Concepts 11g Release 1 (11.1).
- [87]. MARINOV, M. (Jul 2008) Using frames for knowledge representation in a CORBA-based distributed environment. *Know-Based Syst, ACM*, 21, 391-397, URL: <http://dx.doi.org/10.1016/j.knosys.2008.02.003>.
- [88]. MASIF - The Object Management Group's Mobile Agent Systems Interoperability Facility. <http://www.omg.org/> (last visited 2009).
- [89]. MCCABE, T. J. (1976) A Complexity Measure. *IEEE Transactions on Software Engineering*, SE-2, 308-320.
- [90]. MCLEAN, E. R. (1979) End-users as application developers. *MIS Quarterly*, 10(4), 37-46.
- [91]. MCMURTRY, C., MERCURI, M., WATLING, N. & WINKLER, M. (2007) *Windows Communication Foundation Unleashed (Wcf)*, 1st ed, USA: Sam.
- [92]. MCPHEE, D., MHEREEG, M., HOLE, S. & NATHAN, T. (2009) Using a Software Multi-Agent System to Assess Student Learning Outcomes. *4th Plymouth e-learning Conference*. Plymouth, UK.
- [93]. MHEREEG, M., MCPHEE, D., HOLE, S. & ANGEL, P. (2009) Automatic Classification and Graphical Representation of Interactive User Activity via .NET Windows Service Agent Technology. *Fifth International Conference on Autonomic and Autonomous Systems, ICAS*. Spain, IEEE.
- [94]. MICHAEL, L., PETER, M. & CHRIS, P. (2003) Agent technology: Enabling next generation computing: A roadmap for agent-based computing. AgentLink report, available from www.agentlink.org/roadmap, ISBN 0854 327886.
- [95]. MICHELE, L. B. (2007) *Learning WCF*, O'Reilly Media, Inc.
- [96]. Microsoft Corporation, "Comparing Web Service Performance WS Test 1.5 Benchmark Results for .NET 3.5/Windows Server 2008 vs. IBM WebSphere 6.1/Red Hat Linux Advanced Platform 5", © Microsoft Corporation, Feb. 2008.

- URL:<http://download.microsoft.com/download/0/d/6/0d671ad6-bc52-4b50-b7f2-e07000198146/WSTestBenchmark.pdf>.
- [97]. Microsoft Corporation, “Windows Server 2008/.NET Framework 3.5 and IBM WebSphere 6.1 Service-Oriented Performance and Scalability Benchmark”, © Microsoft Corporation, Feb. 2008, URL:
http://download.microsoft.com/download/6/5/0/65012620-40d5-4359-9158-d785147cc034/TradeBenchmark_WinServer2008.pdf.
 - [98]. Microsoft Corporation, Benchmarking IBM WebSphere® 7 on IBM® Power6™ and AIX vs. Microsoft® .NET on Hewlett Packard BladeSystem and Windows Server® 2008. © Microsoft Corporation, March 2009.
 - [99]. MISRA, S. & MISRA, A. K. (2007) Evaluation and comparison of cognitive complexity measure. *ACM Press*.
 - [100]. MITRA, S. & ACHARYA, T. (2003) *Data Mining: Multimedia, Soft Computing, and Bioinformatics*, Hoboken, New Jersey, John Wiley & Sons, Inc.
 - [101]. MONTROSE, C., R (1995) Object-Oriented Development Using The Shlaer-Mellor Method. Paper available from Project Technology, Inc.
 - [102]. MSDN. (2010) Synchronous and Asynchronous Operations.
<http://msdn.microsoft.com/en-us/library/ms734701.aspx>. Microsoft Corporation - MSDN, (Last visited, 2010).
 - [103]. MURCH, R. & JOHNSON, T. (1999) *Intelligent Software Agents.*, Prentice Hall PTR.
 - [104]. MYERS, B. (1986) Visual Programming, Programming By Example, and Program Visualization: A Taxonomy. CHI '86. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM Press.
 - [105]. MYERS, B. A., BURNETT, M. M., WIEDENBECK, S. & KO, A. J. (2007) End User Software Engineering: CHI 2007 special interest group meeting. *CHI '07 extended abstracts on human factors in computing systems*, San Jose, CA, USA, ACM.
 - [106]. NARDI, B., A (1993) *Small Matter of Programming*. MIT Press, Cambridge, MA.
 - [107]. NORD, D., G, MCCABBINS, T., F & NORD, J., H (2006) E-Monitoring in the Workplace: Privacy, Legislation, and Surveillance Software. *ACM Press*, 49, 72-77.
 - [108]. NWANA, H. S. (1996) Software agents: An overview. *Knowledge Engineering Review*, 11(3), 205-244.

- [109]. O'BRIEN LERO, L., MERSON, P. & BASS, L. (2007) Quality Attributes for Service-Oriented Architectures. *International Workshop on Systems Development in SOA Environments, SDSOA '07: ICSE Workshop*.
- [110]. ODELL, J. (1999) Relationship between Agent and Object Technologies. *Agent Technology Green Paper, OMG Agent Work Group*.
- [111]. ODELL, J. (2000) Agents: Technology and usage (Part 1). *Executive Report*, 3(4).
- [112]. OMG Agent Working Group (2000), Agent Technology Green paper, OMG Document ec/2000-08-01, (1).
- [113]. OMG Request for Information, Agent Technology in OMA, OMG Document #ec/99-03-10, <http://www.objs.com/isig/agent-rfi-6.html>, (last visited 2009), August 1999.
- [114]. O'NEAL, M. B. (1993) An empirical study of three common software complexity measures. *Proceedings of the 1993 ACM/SIGAPP symposium on applied computing: states of the art and practice*. Indianapolis, Indiana, United States, ACM Press.
- [115]. Oracle Data Mining,
<http://www.oracle.com/technology/products/bi/quicktour/html/datamining.htm>,
(last visited 2010).
- [116]. OVEIDO, E. I. (1980) Control Flow, Data and Program Complexity. *Proceedings of the IEEE COMPSAC*. Chicago, Illinois, USA, 146-152.
- [117]. PADGHAM, L. & WINIKOFF, M. (2004) *Developing Intelligent Agent Systems: A Practical Guide*, Chichester, Great Britain, Wiley.
- [118]. PAOLO, B., PAOLO, G., FAUSTO, G., MYLOPOULOS, J. & PERINI, A. (2002) Tropos: An agent-oriented software development methodology, Technical Report DIT-02-0015. University of Trento, Department of Information and Communication Technology.
- [119]. PEIRIS, C., MULDER, D., CICORIA, S., BAHREE, A. & PATHAK, N. (2007) *Pro WCF: practical Microsoft SOA implementation*, Apress.
- [120]. POSLAD, S., BUCKLE, P. & HADINGHAM, R. (2000) "Open Source Standards and scalable agencies", Presented at the Autonomous Agents Workshop on Infrastructure of Scalable Multi-Agent Systems. Spain.
- [121]. POSLAD, S., BUCKLE, P. & HADINGHAM, R. (2000) "The FIPA-OS Agent Platform: Open Source for Open Standards". in *Proceedings of PAAM*, Manchester, UK.

- [122]. POWELL, S., NBAKER, K. & LAWSON, B. (2008) A Critical Review of the Literature on Spreadsheet Errors. *Decision Support Systems*, 46, 128-138.
- [123]. PRICE WATERHOUSE AND COOPERS. (2004) The Use of Spreadsheets: Considerations for Section 404 of the Sarbanes-Oxley Act. Price, Waterhouse, Coopers.
- [124]. RAFAEL, H. B. (2006) *programming Multi-Agent Systems*, Utrecht, the Netherlands, Springer.
- [125]. Rational the Software Development Company (2005), Rational Unified Process, Best Practices for Software Development Teams. IBM.
- [126]. REA, S. (2005) *Building Intelligent .NET Applications*, Chichester, Great Britain, Wiley.
- [127]. Resource Description Framework (RDF) schema specification, <http://www.w3.org/TR/WD-RDF-schema>, (Last visited 2009).
- [128]. REYNOLDS, G. (2003) *Ethics in Information Technology*, USA, Thomson.
- [129]. ROBERTO, A. & FLORES-MENDEZ (1999) Standardization of Multi-Agent System Frameworks. *ACM*, 5.
- [130]. ROCKART, J. F. & FLANNERY, L. S. (1983) The management of end-user computing. *Commun. ACM*, 26(10), 776-764.
- [131]. RUTHRUFF, J., CRESWICK, E., BURNETT, M., COOK, C., PRABHAKARARAO, S., FISHER, M. & MAIN, M. (2003) Debugging and Finding Faults: End-User Software Visualizations for Fault Localization. *Proceedings of the 2003 ACM Symposium on Software Visualization*.
- [132]. SCAFFIDI, C., SHAW, M. & MYERS, B. (2005a) The “55M End User Programmers” Estimate Revisited. Technical Report CMUISRI-05-100. Carnegie Mellon University, Pittsburgh, PA.
- [133]. SCAFFIDI, C., SHAW, M. & MYERS, B. (2005b) An Approach for Categorizing End User Programmers to Guide Software Engineering Research. *1st Workshop on End User Software Engineering (WEUSE)*. The International Conference on Software Engineering (ICSE 2005), St. Louis.
- [134]. SCOTT, A. & DELOACH (2001) Analysis and design using MaSE and agent Tool. *In Proceedings of the 12th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS 2001)*.

- [135]. SCOTT, D., MARK, W. & CLINT, S. (2001) Multiagent systems engineering, *International Journal of Software Engineering and Knowledge Engineering*, 11(3), 231-258.
- [136]. SELKER, T. (1994) A Teaching Agent that learns. *Communications of the ACM*, 37 (7), 92-99.
- [137]. SHARP, J. (2007) *Microsoft® Windows® Communication Foundation Step by Step*, Washington: First. Microsoft Press.
- [138]. SHERI, G. (2007) Where is Microsoft Excel Used, http://ezinearticles.com/?expert=Sheri_Graves, (Last Visited, 2010).
- [139]. SHLAER, S. & MELLOR, J., S (1988) Object-Oriented Systems Analysis: Modeling the World in Data. Prentice Hall, Englewood Cliffs, NJ.
- [140]. SHLAER, S. & MELLOR, J., S (1991) *Object Lifecycles: Modeling the World in States*, Prentice Hall, Englewood Cliffs, NJ.
- [141]. SHLAER, S. & MELLOR, J., S (1993) The Shlaer-Mellor Method. paper available from Project Technology, Inc.
- [142]. SHOHAM, Y. (1997) An Overview of Agent-oriented Programming. IN BRADSHAW J, M. (Ed.) *In Software Agents*. Menlo Park, Calif. AAAI Press.
- [143]. SMITH, D. C., CYPHER, A. & SPOHRER, J. (1994) Programming Agents without a programming language. *Communications of the ACM*, 37 (7), 55-67.
- [144]. SYCARA, K. P. (1998) Multi-agent systems. *AI Magazine*.
- [145]. SYCARA, K. P. (1998) The many faces of agents. *AI Magazine*.
- [146]. TAN, P.-N., STEINBACH, M. & KUMAR, V. (2006) *Introduction to Data Mining*, Boston, Pearson Addison Wesley.
- [147]. The European ACTS research programme - "CAMELEON", Performance Assessment Results, Deliverable D9, <http://www.comnets.rwth-aachen.de/~camelon>, (last visited 2009).
- [148]. THUAN, T. & LAM, H. Q. (2003). *NET Framework Essentials*, O'Reilly & Associates, Inc.
- [149]. TSCHANZ, D. W. (2009) *Exchange Server Infrastructure Design: A Service Oriented Approach*, Hobken, New Jersey, John Wiley & Sons, Inc.
- [150]. TSIPTIS, K. & CHORIANOPOULOS, A. (2009) *Data Mining Techniques in CRM: Inside Customer Segmentation*, Chichester - United Kingdom, WILEY.
- [151]. VASTERS, C. (2005) Introduction to Building WCF Services , <http://msdn.microsoft.com/en-us/library/aa480190.aspx>, MSDN (last visited 2010).

- [152]. W3schools.com, Introduction to Microsoft .NET,
http://www.w3schools.com/ngws/ngws_intro.asp, (last visited, 2010).
- [153]. WEIß, G. (2002) Agent orientation in software engineering. *The Knowledge Engineering Review*, 16(4), 349-373.
- [154]. What is End User Computing? Institute of End User Computing,
<http://www.ieuc.org/reception/end-user-computing-defined.html>, (last visited, 2008).
- [155]. WILSON, A., BURNETT, M., BECKWITH, L., GRANATIR, O., CASBURN, L., COOK, C., DURHAM, M. & ROTHERMEL, G. (2003) Harnessing Curiosity to Increase Correctness in End-User Programming. *Proceedings of the Conference on Human Factors in Computing Systems*. ACM Press.
- [156]. Windows Communication Foundation Reliable Sessions, <http://msdn.microsoft.com/en-us/library/ms733136.aspx>, MSDN, (Last Visited 2010).
- [157]. Windows Communication Foundation, MSDN. <http://msdn.microsoft.com/en-us/library/ms735119.aspx> (last visited 2010).
- [158]. Windows Services Hosting, MSDN. http://msdn.microsoft.com/en-us/library/bb332338.aspx#msdnwcfhc_topic4, (last visited, 2010).
- [159]. WOOLDRIDGE, M. & JENNINGS, N. R. (1995) Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2), 115-152.
- [160]. WOOLDRIDGE, M. (1998) Agent-based computing. *Interoperable Communication Networks*, 1(1), 71-97.
- [161]. WOOLDRIDGE, M. (2002) *An Introduction to MultiAgent Systems*, Chichester, UK, John Wiley & Sons.
- [162]. WOOLDRIDGE, M., JENNINGS, N. R. & KINNY, D. (2000) The Gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3(3).
- [163]. XINDONG, W., VIPIN, K., ROSS, J., JOYDEEP, G., QIANG, Y., HIROSHI, M., GEOFFREY, J., ANGUS, N., BING, L., PHILIP, S., ZHI-HUA, Z., MICHAEL, S., DAVID, J. & DAN, S. (2007) Top 10 algorithms in data mining. *IEEE International Conference on Data Mining (ICDM)*. Springer.
- [164]. XML-Data, <http://www.w3.org/TR/1998/NOTE-XML-data>, (Last visited 2009).

Appendices

Appendix A: Agent Properties

Property	Definition	Source
Autonomy	It means that the agent can act without direct intervention by humans or other agents and that it has control over its own actions and internal state.	[144, 145]
Reactivity or Situatedness	It means that the agent receives some form of sensory input from its environment, and it performs some action that changes its environment in some way).	[21, 144, 145]
Learning or adaptivity	It means that an agent is capable of i) reacting flexibly to changes in its environment; ii) taking goal-directed initiative, when appropriate; and iii) learning from its own experience, its environment, and interactions with others.	
Proactiveness	It means that the agent does not simply act in response to its environment; it is able to exhibit goal-directed behaviour by taking the initiative.	[21, 111, 159]
Flexibility	It means that the system is responsive (the agents should perceive their environment and respond in a timely fashion to changes that occur in it), pro-active and social.	[68, 69]
Mobility	It means that the agent is able to transport itself from one machine to another and across different system architectures and platforms.	[32]
Temporal continuity	It means that the agent is a continuously running process, not a "one-shot" computation that maps a single input to a single output, then terminates.	
Personality or character	An agent has a well-defined, believable "personality" and emotional state.	
Reusability	Processes or subsequent instances can require keeping instances of the class 'agent' for an information handover or to check and to analyze them according to their results.	[58]
Resource limitation	An agent can only act as long as it has resources at its disposal. These resources are changed by its acting and possibly also by delegating.	
Veracity	It is the assumption that an agent will not knowingly	

	communicate false information.	[159, 160]
Benevolence	It is the assumption that agents do not have conflicting goals and that every agent will therefore always try to do what is asked of it.	
Rationality	It is the assumption that an agent will act in order to achieve its goals, and will not act in such a way as to prevent its goals being achieved — at least insofar as its beliefs permit.	
Inferential capability	An agent can act on abstract task specification using prior knowledge of general goals and preferred methods to achieve flexibility; goes beyond the information given, and may have explicit models of self, user, situation, and/or other agents.	[14]
“Knowledge-level” communication ability	The ability to communicate with persons and other agents with language more resembling humanlike “speech acts” than typical symbol-level program-to-program protocols.	
Prediction ability	An agent is predictive if its model of how the world works is sufficiently accurate to allow it to correctly predict how it can achieve the task.	[42]
Interpretation ability	An agent is interpretive if can correctly interpret its sensor readings.	
Sound	An agent is sound if it is predictive, interpretive and rational.	
Social ability	It means that the agent interacts and this interaction is marked by friendliness or pleasant social relations; that is, the agent is affable, companionable or friendly.	[111]
Coordination	It means that the agent is able to perform some activity in a shared environment with other agents. Activities are often coordinated via plans, workflows, or some other process management mechanism.	
Cooperation or collaboration	It means that the agent is able to coordinate with other agents to achieve a common purpose; non-antagonistic agents that succeed or fail together.	
Proxy ability	An agent can act on behalf of someone or something that is, acting in the interest of, as a representative of, or for	

	the benefit of, some entity.	
Intelligence	The agent's state is formalized by knowledge and interacts with other agents using symbolic language.	
Unpredictability	An agent is able to act in ways that are not fully predictable, even if all the initial conditions are known. It is capable of nondeterministic behaviour.	
Credibility	An agent has a believable personality and emotional state.	
Transparency and accountability	An agent must be transparent when required, but must provide a log of its activities upon demand.	
Competitiveness	An agent is able to coordinate with other agents except that the success of one agent implies the failure of others.	
Ruggedization	An agent is able to deal with errors and incomplete data robustly.	
Trustworthiness	An agent adheres to laws of robotics and is truthful.	

Appendix B: The MA's Code

Appendix B1: FileProperties.cs Code

```
public class FileProperties
{
    private DSOFile.SummaryProperties dsoSummProperties;
    private string docFileToRead;
    private string docAuthor;
    private string docLastSavedBy;
    private string docTitle;
    private string docSubject;
    private string docCompany;
    private string docComments;
    private string docApplication;
    private string docVersion;
    private string docDateCreated = null;
    private string docDateLastSaved = null;
    private string docDocumentName;
    private string docPathName;
    private DSOFile.OleDocumentProperties dsoDocument;

    internal string Author
    {
        get { return docAuthor; }
        set { docAuthor = Author; }
    }
    internal string LastSavedBy
    {
        get { return docLastSavedBy; }
        set { docLastSavedBy = LastSavedBy; }
    }
    internal string Title
    {
        get { return docTitle; }
        set { docTitle = Title; }
    }
    internal string Subject
    {
        get { return docSubject; }
        set { docSubject = Subject; }
    }
    internal string Company
    {
        get { return docCompany; }
        set { docCompany = Company; }
    }
    internal string Comments
    {
        get { return docComments; }
        set { docComments = Comments; }
    }
    internal string AppName
    {
        get { return docApplication; }
        set { docApplication = AppName; }
    }
    internal string Version
    {
        get { return docVersion; }
    }
}
```



```

        set { docVersion = Version; }
    }
    internal string DateCreated
    {
        get { return docDateCreated; }
        set { docDateCreated = DateCreated; }
    }
    internal string DateLastSaved
    {
        get { return docDateLastSaved; }
        set { docDateLastSaved = DateLastSaved; }
    }
    internal string DocumentName
    {
        get { return docDocumentName; }
        set { docDocumentName = DocumentName; }
    }
    internal string DocumentPath
    {
        get { return docPathName; }
        set { docPathName = DocumentPath; }
    }
}

private void ReadFileProperties(string FileName)
{
    //Create a new Excel based document and open the target file
    dsoDocument = new DSOFile.OleDocumentProperties();
    dsoDocument.Open(FileName, true,
        DSOFile.dsoFileOpenOptions.dsoOptionOpenReadOnlyIfNoWriteAccess);

    //Create the file summary properties
    dsoSummProperties = dsoDocument.SummaryProperties;

    //Read the summary properties
    docAuthor = dsoSummProperties.Author;
    docLastSavedBy = dsoSummProperties.LastSavedBy;

    docApplication = dsoSummProperties.ApplicationName;
    docVersion = dsoSummProperties.Version;
    docDateCreated = dsoSummProperties.DateCreated.ToString();
    docDateLastSaved = dsoSummProperties.DateLastSaved.ToString();

    //Read the document properties
    docDocumentName = dsoDocument.Name;
    docPathName = dsoDocument.Path;

    if (dsoSummProperties.Title == null)
    {
        docTitle = "Not Set";
    }
    else
    {
        docTitle = dsoSummProperties.Title;
    }
    if (dsoSummProperties.Subject == null)
    {
        docSubject = "Not Set";
    }
    else
    {
        docSubject = dsoSummProperties.Subject;
    }
}

```

```

    }
    if (dsoSummProperties.Company == null)
    {
        docCompany = "Not Set";
    }
    else
    {
        docCompany = dsoSummProperties.Company;
    }
    if (dsoSummProperties.Comments == null)
    {
        docComments = "Not Set";
    }
    else
    {
        docComments = dsoSummProperties.Comments;
    }
}

public FileProperties(string Filename)
{
    docFileToRead = Filename;
    ReadFileProperties(docFileToRead);
}
}

```

Appendix B2: Functions.cs Code

```
using System.Runtime.InteropServices;
using Microsoft.Vbe.Interop;
using Microsoft.Office.Interop;
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Globalization;
using System.IO;
using System.Text;
using System.Text.RegularExpressions;
using System.Data.SqlClient;
using System.Xml;
using Microsoft.Office.Interop.Excel;
using Excel = Microsoft.Office.Interop.Excel;

public class Functions
{
    // Date and Time property procedures
    private int dtDate;
    private int dtDateValue;
    private int dtDay;
    private int dtDays360;
    private int dtHour;
    private int dtMinute;
    private int dtMonth;
    private int dtNow;
    private int dtSecond;
    private int dtTime;
    private int dtTimeValue;
    private int dtToday;
    private int dtWeekday;
    private int dtYear;
    // Database Property Procedures
    private int dbDAverage;
    private int dbDCount;
    private int dbDcountA;
    private int dbDGet;
    private int dbDMax;
    private int dbDMin;
    private int dbDProduct;
    private int dbDStdev;
    private int dbDStdevP;
    private int dbDSum;
    private int dbDVar;
    private int dbDVarp;
    //Grouped Variable Definition
    private int DateAndTimeFunc;
    private int LookAndRefFunc;
    private int DatabaseFunc;
    private int TextFunc;
    private int LogicalFunc;
    private int InfoFunc;
    private int FinancialFunc;
    private int MathAndTrigFunc;
    private int StatFunc;
    private int CubeFunc;
    private int EngineeringFunc;
    private int CppFunc;
```

```

.....
.....

// Date and Time property procedures
public int DateFormula
{
    get { return dtDate; }
    set { dtDate = DateFormula; }
}
public int DateValue
{
    get { return dtDateValue; }
    set { dtDateValue = DateValue; }
}
public int day
{
    get { return dtDay; }
    set { dtDay = day; }
}
public int Days360
{
    get { return dtDays360; }
    set { dtDays360 = Days360; }
}
public int Hour
{
    get { return dtHour; }
    set { dtHour = Hour; }
}
public int Minute
{
    get { return dtMinute; }
    set { dtMinute = Minute; }
}
public int Month
{
    get { return dtMonth; }
    set { dtMonth = Month; }
}
public int Now
{
    get { return dtNow; }
    set { dtNow = Now; }
}
public int Second
{
    get { return dtSecond; }
    set { dtSecond = Second; }
}
public int Time
{
    get { return dtTime; }
    set { dtTime = Time; }
}
public int TimeValue
{
    get { return dtTimeValue; }
    set { dtTimeValue = TimeValue; }
}
public int Today
{

```

```

        get { return dtToday; }
        set { dtToday = Today; }
    }
    public int WeekDay
    {
        get { return dtWeekday; }
        set { dtWeekday = WeekDay; }
    }
    public int Year
    {
        get { return dtYear; }
        set { dtYear = Year; }
    }

    // Database Property Procedures
    public int DAverage
    {
        get { return dbDAverage; }
        set { dbDAverage = DAverage; }
    }
    public int DCount
    {
        get { return dbDCount; }
        set { dbDCount = DCount; }
    }
    public int DCountA
    {
        get { return dbDcountA; }
        set { dbDcountA = DCountA; }
    }
    public int DGet
    {
        get { return dbDGet; }
        set { dbDGet = DGet; }
    }
    public int DMax
    {
        get { return dbDMax; }
        set { dbDMax = DMax; }
    }
    public int DMin
    {
        get { return dbDMin; }
        set { dbDMin = DMin; }
    }
    public int DProduct
    {
        get { return dbDProduct; }
        set { dbDProduct = DProduct; }
    }
    public int DStDev
    {
        get { return dbDStdev; }
        set { dbDStdev = DStDev; }
    }
    public int DStDevP
    {
        get { return dbDStdevP; }
        set { dbDStdevP = DStDevP; }
    }
    public int DSum

```

```

{
    get { return dbDSum; }
    set { dbDSum = DSum; }
}
public int DVar
{
    get { return dbDVar; }
    set { dbDVar = DVar; }
}
public int DVarP
{
    get { return dbDVarp; }
    set { dbDVarp = DVarP; }
}

// Grouped Variable Definition
public int FreqFunc
{
    get { return FrequentFunc; }
    set { FrequentFunc = FreqFunc; }
}
public int DandTFunc
{
    get { return DateAndTimeFunc; }
    set { DateAndTimeFunc = DandTFunc; }
}
public int LandRFunc
{
    get { return LookAndRefFunc; }
    set { LookAndRefFunc = LandRFunc; }
}
public int DBFunc
{
    get { return DatabaseFunc; }
    set { DatabaseFunc = DBFunc; }
}
public int TFunc
{
    get { return TextFunc; }
    set { TextFunc = TFunc; }
}
public int LFunc
{
    get { return LogicalFunc; }
    set { LogicalFunc = LFunc; }
}
public int IFunc
{
    get { return InfoFunc; }
    set { InfoFunc = IFunc; }
}
public int FFunc
{
    get { return FinancialFunc; }
    set { FinancialFunc = FFunc; }
}
public int MandTFunc
{
    get { return MathAndTrigFunc; }
    set { MathAndTrigFunc = MandTFunc; }
}

```

```

public int SFunc
{
    get { return StatFunc; }
    set { StatFunc = SFunc; }
}
public int CFunc
{
    get { return CubeFunc; }
    set { CubeFunc = CFunc; }
}
public int EFunc
{
    get { return EngineeringFunc; }
    set { EngineeringFunc = EFunc; }
}
public int AAFunc
{
    get { return AddInsAutoFunc; }
    set { AddInsAutoFunc = AAFunc; }
}
public int CpFunc
{
    get { return CppFunc; }
    set { CppFunc = CpFunc; }
}
public Functions(string FileName)
{
    ReadFile(FileName);
}

.....
.....
.....

private Excel.Application ExcelObjf = null;
private void ReadFile(string FileName)
{
    try
    {
        // Construct an Excel Application Object
        ExcelObjf = new Excel.Application();

        // Here is the call to Open a Workbook in Excel
        Excel.Workbook theWorkbook = ExcelObjf.Workbooks.
        Open(FileName.Trim(), 0, true, 5, "", "", true,
        Excel.XlPlatform.xlWindows, "\t", false, false, 0, true, 1, 0);

        // Get the collection of sheets in the workbook
        Excel.Sheets sheets = theWorkbook.Worksheets;
        Excel.Worksheet workSheet = Excel.Worksheet)
        theWorkbook.ActiveSheet;

        foreach (Excel.Worksheet Sheet in sheets)
        {
            for (int i = 1; i <= 300; i++)
            {
                Excel.Range Range = Sheet.get_Range("A" + i.ToString(), "IV" +
                i.ToString());
                System.Array mValues= System.Array)Range.Cells.Formula;
                int ArrayLen = mValues.Length - 1;
                string[] ExcelArray = new string[ArrayLen + 1];
            }
        }
    }
}

```

```

        string[] ExcelArray1 = new string[ArrayLen + 1];
        ExcelArray1 = ConvertToStringArray(mValues);
    }
}
// Clean up and exit.
TheWorkBook = null;
}
catch (NullReferenceException nre)
{
    Console.WriteLine(nre.ToString());
}
}

private string[] ConvertToStringArray(System.Array Values)
{
    int length = Values.Length;
    string[] RowArray = new string[length];

    for (int LoopCount = 1; LoopCount <= length; LoopCount++)
    {
        //MessageBox.Show(Values.GetValue(1, LoopCount))
        if (Values.GetValue(1, LoopCount).ToString() == "")
        {
            RowArray[LoopCount - 1] = "empty";
        }
        else
        {
            RowArray[LoopCount - 1] = Values.GetValue(1,
                LoopCount).ToString();
            //Check for a formula
            string CellContent = RowArray[LoopCount - 1].ToString();
            if (CellContent.Substring(0, 1) == "=")
            {
                FormulaFound(CellContent);
            }
        }
    }
    return RowArray;
}
private void FormulaFound(string Formula)
{
    FormulaMatch(Formula);
}

public void FormulaMatch(string Formula)
{
    TextInfo myTI = new CultureInfo("en-US", false).TextInfo;
    string UCFormula = myTI.ToUpper(Formula);

.....
.....
.....

//Date and Time Functions
    else if
    ((Microsoft.VisualBasic.CompilerServices.StringType.StrLike(UCFormula,
    "=DATE(*)", Microsoft.VisualBasic.CompareMethod.Binary)))
    {
        dtDate += 1;
        DateAndTimeFunc += 1;
    }
}

```



```

    }
    else if
    ((Microsoft.VisualBasic.CompilerServices.StringType.StrLike(UCFormula,
    "=DATEVALUE(*)", Microsoft.VisualBasic.CompareMethod.Binary)))
    {
        dtDateValue += 1;
        DateAndTimeFunc += 1;
    }
    else if
    ((Microsoft.VisualBasic.CompilerServices.StringType.StrLike(UCFormula,
    "=DAY(*)", Microsoft.VisualBasic.CompareMethod.Binary)))
    {
        dtDay += 1;
        DateAndTimeFunc += 1;
    }
    else if
    ((Microsoft.VisualBasic.CompilerServices.StringType.StrLike(UCFormula,
    "=DAYS360(*)", Microsoft.VisualBasic.CompareMethod.Binary)))
    {
        dtDays360 += 1;
        DateAndTimeFunc += 1;
    }
    else if
    ((Microsoft.VisualBasic.CompilerServices.StringType.StrLike(UCFormula,
    "=HOUR(*)", Microsoft.VisualBasic.CompareMethod.Binary)))
    {
        dtHour += 1;
        DateAndTimeFunc += 1;
    }
    else if
    ((Microsoft.VisualBasic.CompilerServices.StringType.StrLike(UCFormula,
    "=MINUTE(*)", Microsoft.VisualBasic.CompareMethod.Binary)))
    {
        dtMinute += 1;
        DateAndTimeFunc += 1;
    }
    else if
    ((Microsoft.VisualBasic.CompilerServices.StringType.StrLike(UCFormula,
    "=MONTH(*)", Microsoft.VisualBasic.CompareMethod.Binary)))
    {
        dtMonth += 1;
        DateAndTimeFunc += 1;
    }
    else if
    ((Microsoft.VisualBasic.CompilerServices.StringType.StrLike(UCFormula,
    "=NOW(*)", Microsoft.VisualBasic.CompareMethod.Binary)))
    {
        dtNow += 1;
        DateAndTimeFunc += 1;
    }
    else if
    ((Microsoft.VisualBasic.CompilerServices.StringType.StrLike(UCFormula,
    "=SECOND(*)", Microsoft.VisualBasic.CompareMethod.Binary)))
    {
        dtSecond += 1;
        DateAndTimeFunc += 1;
    }
    else if
    ((Microsoft.VisualBasic.CompilerServices.StringType.StrLike(UCFormula,
    "=TIME(*)", Microsoft.VisualBasic.CompareMethod.Binary)))
    {

```

```

        dtTime += 1;
        DateAndTimeFunc += 1;
    }
    else if
    ((Microsoft.VisualBasic.CompilerServices.StringType.StrLike(UCFormula,
    "=TIMEVALUE(*)", Microsoft.VisualBasic.CompareMethod.Binary)))
    {
        dtTimeValue += 1;
        DateAndTimeFunc += 1;
    }
    else if
    ((Microsoft.VisualBasic.CompilerServices.StringType.StrLike(UCFormula,
    "=TODAY(*)", Microsoft.VisualBasic.CompareMethod.Binary)))
    {
        dtToday += 1;
        DateAndTimeFunc += 1;
    }
    else if
    ((Microsoft.VisualBasic.CompilerServices.StringType.StrLike(UCFormula,
    "=WEEKDAY(*)", Microsoft.VisualBasic.CompareMethod.Binary)))
    {
        dtWeekday += 1;
        DateAndTimeFunc += 1;
    }
    else if
    ((Microsoft.VisualBasic.CompilerServices.StringType.StrLike(UCFormula,
    "=YEAR(*)", Microsoft.VisualBasic.CompareMethod.Binary)))
    {
        dtYear += 1;
        DateAndTimeFunc += 1;
    }

    // Database Functions
    else if
    ((Microsoft.VisualBasic.CompilerServices.StringType.StrLike(UCFormula,
    "=DAVERAGE(*)", Microsoft.VisualBasic.CompareMethod.Binary)))
    {
        dbDAverage += 1;
        DatabaseFunc += 1;
    }
    else if
    ((Microsoft.VisualBasic.CompilerServices.StringType.StrLike(UCFormula,
    "=DCOUNT(*)", Microsoft.VisualBasic.CompareMethod.Binary)))
    {
        dbDCount += 1;
        DatabaseFunc += 1;
    }
    else if
    ((Microsoft.VisualBasic.CompilerServices.StringType.StrLike(UCFormula,
    "=DCOUNTA(*)", Microsoft.VisualBasic.CompareMethod.Binary)))
    {
        dbDcountA += 1;
        DatabaseFunc += 1;
    }
    else if
    ((Microsoft.VisualBasic.CompilerServices.StringType.StrLike(UCFormula,
    "=DGET(*)", Microsoft.VisualBasic.CompareMethod.Binary)))
    {
        dbDGet += 1;
        DatabaseFunc += 1;
    }

```

```

        else if
        ((Microsoft.VisualBasic.CompilerServices.StringType.StrLike(UCFormula,
"=DMAX(*)", Microsoft.VisualBasic.CompareMethod.Binary)))
        {
            dbDMax += 1;
            DatabaseFunc += 1;
        }
        else if
        ((Microsoft.VisualBasic.CompilerServices.StringType.StrLike(UCFormula,
"=DMIN(*)", Microsoft.VisualBasic.CompareMethod.Binary)))
        {
            dbDMin += 1;
            DatabaseFunc += 1;
        }
        else if
        ((Microsoft.VisualBasic.CompilerServices.StringType.StrLike(UCFormula,
"=DPRODUCT(*)", Microsoft.VisualBasic.CompareMethod.Binary)))
        {
            dbDProduct += 1;
            DatabaseFunc += 1;
        }
        else if
        ((Microsoft.VisualBasic.CompilerServices.StringType.StrLike(UCFormula,
"=DSTDEV(*)", Microsoft.VisualBasic.CompareMethod.Binary)))
        {
            dbDStdev += 1;
            DatabaseFunc += 1;
        }
        else if
        ((Microsoft.VisualBasic.CompilerServices.StringType.StrLike(UCFormula,
"=DSTDEVP(*)", Microsoft.VisualBasic.CompareMethod.Binary)))
        {
            dbDStdevP += 1;
            DatabaseFunc += 1;
        }
        else if
        ((Microsoft.VisualBasic.CompilerServices.StringType.StrLike(UCFormula,
"=DSUM(*)", Microsoft.VisualBasic.CompareMethod.Binary)))
        {
            dbDSum += 1;
            DatabaseFunc += 1;
        }
        else if
        ((Microsoft.VisualBasic.CompilerServices.StringType.StrLike(UCFormula,
"=DVAR(*)", Microsoft.VisualBasic.CompareMethod.Binary)))
        {
            dbDVar += 1;
            DatabaseFunc += 1;
        }
        else if
        ((Microsoft.VisualBasic.CompilerServices.StringType.StrLike(UCFormula,
"=DVARP(*)", Microsoft.VisualBasic.CompareMethod.Binary)))
        {
            dbDVarp += 1;
            DatabaseFunc += 1;
        }
    }
}

```

Appendix B3: CodeConstructs.cs Code

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Data;
using System.Diagnostics;
using System.ComponentModel;
using System.Runtime.InteropServices;
using Microsoft.Vbe.Interop;
using Microsoft.Office.Interop;
using System.Text;
using System.Text.RegularExpressions;
using Microsoft.Office.Interop.Excel;
using Excel = Microsoft.Office.Interop.Excel;

public class CodeConstructs
{
    private string ExcelFileName;
    private string strVBACode;
    private RegularExpressions myRegularExpression = new
    RegularExpressions();
    public ArrayList ConstructList = new ArrayList();
    private ArrayList MatchList = new ArrayList();
    private string MatchCount;
    private int privateSub;
    private int publicSub;
    private int privateFunction;
    private int publicFunction;
    private int asNew;
    private int forEach;
    private int doLoops;

    public int privateSubCount
    {
        get { return privateSub; }
        set { privateSub = privateSubCount; }
    }
    public int publicSubCount
    {
        get { return publicSub; }
        set { publicSub = publicSubCount; }
    }

    .....
    .....

    public int forEachCount
    {
        get { return forEach; }
        set { forEach = forEachCount; }
    }
    public int doLoopsCount
    {
        get { return doLoops; }
        set { doLoops = doLoopsCount; }
    }
    private void ReadExcelVBACode(string FileName)
    {
        // Declare variables to access the Excel workbook.
        Excel.Application objXLApp = null;
```

```

Microsoft.Office.Interop.Excel.Workbooks objXLWorkbooks = null;
Microsoft.Office.Interop.Excel.Workbook objXLABC = null;
// Declare variables to access the macros in the workbook.
VBProject objProject = null;
CodeModule objCode = null;

try
{
    // Declare other miscellaneous variables.
    int iLine = 0;
    // Open Excel, and open the workbook.
    objXLApp = new Excel.Application();
    objXLWorkbooks = objXLApp.Workbooks;
    objXLABC = objXLWorkbooks.Open(FileName, 0, true, 5, "", "",
    true, Excel.XlPlatform.xlWindows, "\t", false, false, 0, true, 1, 0);
    // Get the project details in the workbook.
    objProject = objXLABC.VBProject;
    // Iterate through each component in the project.
    foreach (VBComponent objComponent in objProject.VBComponents)
    {
        // Find the code module for the project.
        objCode = objComponent.CodeModule;
        // Scan through the code module, looking for procedures.
        iLine = 1;
        while (iLine < objCode.CountOfLines)
        {
            strVBACode = strVBACode + objCode.get_Lines(iLine,
            1) + System.Environment.NewLine;
            iLine = iLine + 1;
        }
        objCode = null;
    }
    objProject = null;

    // Clean up and exit.
    objXLApp.Quit();
}
catch (NullReferenceException nre)
{
    Console.WriteLine(nre.ToString());
}
}

public CodeConstructs(string ExcelName)
{
    strVBACode = null;
    ExcelFileName = ExcelName;
    ReadExcelVBACode(ExcelFileName);
    SearchMacroCode();
}

private void SearchMacroCode()
{
    string Source = VBACode;
    string RegExpression;

    RegExpression = myRegularExpression.PrivateSubRoutine;
    privateSub = SearchCode(RegExpression, Source, "Private Sub");

    RegExpression = myRegularExpression.PublicSubRoutine;
    publicSub = SearchCode(RegExpression, Source, "Public Sub");
}

```

```

.....
.....

RegularExpression = myRegularExpression.ForEachConstruct;
forEach = SearchCode(RegularExpression, Source, "For Each");

RegularExpression = myRegularExpression.DoLoopConstruct;
doLoops = SearchCode(RegularExpression, Source, "Do Loops");

}

private int SearchCode(string RegExp, string Code, string Construct)
{
    int functionReturnValue = 0;
    try
    {
        Regex re = new Regex(RegExp, RegexOptions.IgnoreCase);
        MatchCollection mc = re.Matches(Code);
        MatchCount = mc.Count.ToString();
        foreach (Match m in re.Matches(Code))
        {
            MatchList.Add(m.ToString());
        }
        myRegularExpression.MatchCount(mc.Count, Construct);
        functionReturnValue = mc.Count;
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.ToString());
    }
    return functionReturnValue;
}
}

```

Appendix B4: RegularExpressions.cs Code

```
public class RegularExpressions
{
    private const string PublicSub =
    "(?<=(?:\\n|:|^)\\s*?) (Public\\sSub.*)";
    private const string PrivateSub =
    "(?<=(?:\\n|:|^)\\s*?) (Private\\sSub.*)";
    private const string PublicFunction =
    "(?<=(?:\\n|:|^)\\s*?) (Public\\sFunction.*)";
    private const string PrivateFunction =
    "(?<=(?:\\n|:|^)\\s*?) (Private\\sFunction.*)";
    private const string DimAsNew = ("Dim.*As.*New.*");
    private const string AsNew =
    ("Private.*As\\sNew.*|Public.*As\\sNew.*|Friend.*As\\sNew.*");
    private const string ForEach = ("For\\sEach.*");
    private const string DoLoops =
    ("Do\\sWhile.*|Do\\sUntil.*|Loop\\sWhile.*|Loop\\sUntil.*");
    private int PublicSubCount;
    private int PrivateSubCount;
    private int PublicFunctionCount;
    private int PrivateFunctionCount;
    private int AsNewCount;
    private int ForEachCount;
    private int DoLoopsCount;

    public string PublicSubRoutine
    {
        get { return PublicSub; }
        set { PublicSubRoutine = PublicSub; }
    }
    public string PrivateSubRoutine
    {
        get { return PrivateSub; }
        set { PrivateSubRoutine = PrivateSub; }
    }
    .....
    .....
    public string ForEachConstruct
    {
        get { return ForEach; }
        set { ForEachConstruct = ForEach; }
    }
    public string DoLoopConstruct
    {
        get { return DoLoops; }
        set { DoLoopConstruct = DoLoops; }
    }
    public void MatchCount(int Count, string Construct)
    {
        switch (Construct)
        {
            case "Public Sub":
                PublicSubCount = Count;
                break;
            case "Private Sub":
                PrivateSubCount = Count;
                break;
            .....
            .....
            case "For Each":
```

```
        ForEachCount = Count;
        break;
    case "Do Loops":
        DoLoopsCount = Count;
        break;
    }
}
```


Appendix B5: app.config Configuration File

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>

  <system.serviceModel>
    <services>
      <service
        name="AutoService_StreamTCP_MiniNet.ServiceModel.Agents.AutomaticService_StreamTCP_MiniNet"
        behaviorConfiguration="CalculatorServiceBehavior">
        <host>
          <baseAddresses>
            <add baseAddress="http://J345-MM.uni.glam.ac.uk:9000/ServiceModelAgents/Service"/>
          </baseAddresses>
        </host>

        <!-- this endpoint is exposed at:
        net.tcp://localhost:9000/servicemodelsamples/service -->
        <endpoint address="net.tcp://J345-MM.uni.glam.ac.uk:8000/ServiceModelAgents/Service"
          binding="netTcpBinding"
          bindingConfiguration="StreamedTCPBinding"
          contract="AutoService_StreamTCP_MiniNet.ServiceModel.Agents.IAutomation_StreamTCP_MiniNet" />
        <!-- the mex endpoint is exposed at
        http://localhost:8000/ServiceModelSamples/service/mex -->
        <endpoint address="mex"
          binding="mexHttpBinding"
          contract="IMetadataExchange" />
      </service>
    </services>

    <bindings>
      <!--
        Following is the expanded configuration section for a
        NetTcpBinding.
        Each property is configured with the default values.
        See the Message Security, and Transport Security samples in the
        WS binding samples to learn how to configure these features.
      -->
      <netTcpBinding>
        <binding name="StreamedTCPBinding"
          closeTimeout="00:01:00"
          openTimeout="00:01:00"
          receiveTimeout="00:10:00"
          sendTimeout="00:01:00"
          transactionFlow="false"
          transferMode="StreamedResponse"
          transactionProtocol="OleTransactions"
          hostNameComparisonMode="StrongWildcard"
          listenBacklog="10"
          maxBufferPoolSize="524288"
          maxBufferSize="65536"
          maxConnections="10"
          maxReceivedMessageSize="204003200">
          <readerQuotas maxDepth="32"
            maxStringContentLength="8192"
            maxArrayLength="16384"
            maxBytesPerRead="4096"

```

```

        maxNameTableCharCount="16384" />
    <reliableSession ordered="true"
        inactivityTimeout="00:10:00"
        enabled="false" />
    <security mode="Transport">
        <transport clientCredentialType="Windows"
            protectionLevel="EncryptAndSign" />
    </security>
</binding>
</netTcpBinding>
</bindings>

<!--For debugging purposes set the includeExceptionDetailInFaults
attribute to true-->
<behaviors>
    <serviceBehaviors>
        <behavior name="CalculatorServiceBehavior">
            <serviceMetadata httpGetEnabled="true" />
            <serviceDebug includeExceptionDetailInFaults="False" />
        </behavior>
    </serviceBehaviors>
</behaviors>

</system.serviceModel>
</configuration>

```

Appendix C: Excel Spreadsheet Components/Features Groupings

Component		Features
File properties		Author, LastSavedBy, Title, Subject, Company, Comments, AppName, Version, DateCreated, DateLastSaved, DocumentName, DocumentPath
Functions	Database	DAVERAGE, DCOUNT, DCOUNTA, DGET, DMAX, DMIN, DPRODUCT, DSTDEV, DSTDEVP, DSUM, DVAR, DVARP
	Lookup & Reference	ADDRESS, AREAS, CHOOSE, COLUMN, COLUMNS, HLOOKUP, HYPERLINK, INDEX, LOOKUP, MATCH, OFFSET, ROW, ROWS, TRANSPOSE, VLOOKUP, INDIRECT
	Math & trigonometry functions	ABS, ACOS, ACOSH, ASIN, ASINH, ATAN, ATAN2, ATANH, CEILING, COMBIN, COS, COSH, DEGREES, EVEN, EXP, FACT, FLOOR, INT, LN, LOG, LOGIO, MDETERM, MINVERSE, MMULT, MOD, ODD, PI, POWER, PRODUCT, RADIANS, RAND, ROMAN, ROUND, ROUNDDOWN, ROUNDUP, SIGN, SIN, SINH, SQRT, SUBTOTAL, SUMIF, SUMPRODUCT, SUMSQ, SUMX2MY2, SUMX2PY2, SUMXMY2, TAN, TANH, TRUNC
	Statistical Functions	AVEDEV, AVERAGEA, BETADIST, BETAINV, BINOMDIST, CHIDIST, CHIINV, CHITEST, CONFIDENCE, CORREL, COUNTA, COUNTBLANK, COVAR, CRITBINOM, DEVSQ, EXPONDIST, FDIST, FINV, FISHER, FISHERINV, FORECAST, FREQUENCY, GAMMADIST, GAMMAINV, GAMMALN, GEOMEAN, GROWTH, HARMEAN, HYGEOMDIST, INTERCEPT, KURT, LARGE, LINEST, LOGEST, LOGINV, LOGNORMDIST, MAXA, MINA, NEGBINOMDIST, NORMDIST, NORMEINV, PEARSON, PERCENTILE, PERCENTRANK, PERMUT, POISSON, PROB, QUARTILE, RANK, RSQ, SKEW, SLOPE, SMALL, STANDARDIZE, STDEVA, STEDP, STEDPA, STEYX, TDIST, TINV, TREND, TRIMMEAN, TTEST, VAR, VARA, VARP, VARPA, WEIBULL, ZTEST
	Financial	DB, DDB, SLN, SYD, VDB, AMORLINC, AMORDEGRC, CUMIPMT, CUMPRINC, EFFECT, FV, FVSCHEDULE, IPMT, IRR, ISPMT, MIRR, NPER, NPV, PMT, PPMT, PV, RATE, XIRR, XNPV, COUPDAYBS, COUPDAYS, COUPDAYSSNC, COUPNCD, COUPPCD, COUPNUM, ACCRINT, ACCRINTM, DISC, DURATION, INTRATE, MDURATION, NOMINAL, ODPRICE, ODYIELD, ODDLPRICE, ODDLYIELD, PRICE, PRICEDISC, PRICEMAT, RECEIVED, YIELD, YIELDDISC, YIELDMAT, DOLLARDE, DOLLARFR, TBILLEQ, TBILLPRICE, TBILLYIELD
	Date & Time	DATE, DATEVALUE, DAY, DAYS360, HOUR, MINUTE, MONTH, NOW, SECOND, TIME, TIMEVALUE, TODAY, WEEKDAY, YEAR
	Text	CHAR, CLEAN, CODE, CONCATENATE, DOLLAR, EXACT, FIND, FIXED, LEFT, LEN, LOWER, MID, PROPER, REPLACE, REPT, RIGHT, SEARCH, SUBSTITUTE, T, TEXT, TRIM, UPPER, VALUE
	Information	CELL, ERRORTYPE, INFO, ISBLANK, ISERR, ISLOGICAL, ISNA, ISNONTEXT,

		ISNUMBER
	Logical	AND, FALSE, IF, NOT, OR, TRUE
	Cube	CUBEKPIMEMBER, CUBEMEMBER, CUBEMEMBERPROPERTY, CUBERANKEDMEMBER, CUBESET, CUBESETCOUNT;
	Engineering	BESSELI, BESSELJ, BESSELK, BESSELY, BIN2DEC, BIN2HEX, BIN2OCT, COMPLEX, CONVERT, DEC2BIN, DEC2HEX, DEC2OCT, DELTA, ERF, ERF, GESTEP, HEX2BIN, HEX2DEC, HEX2OCT, IMABS, IMAGINARY, IMARGUMENT, IMCONJUGATE, IMCOS, IMDIV, IMEXP, IMLN, IMLOG10, IMLOG2, IMPOWER, IMPRODUCT, IMREAL, IMSIN, IMSQRT, IMSUB, IMSUM, OCT2BIN, OCT2DEC, OCT2HEX
	Add-ins and Automation	EUROCONVERTER, GETPIVOTDATA, REGISTER_ID, SQL_REQUEST
Applications & Languages	ActiveX Controls	EMBED(*)
	Automated applications	Add-ins and Automation functions
	ADO	Database functions
	C/C++	Call(), Result(), Return()
	VB6 or VB.net	Register(), Unregister()
Code Constructs	Iterations	Do ... Loop Until, Do ... Until Loop, Do... Loop While, Do ...While Loop, For ... Next, For ... Each
	Selections	If...Then...End If, If...Then...Else... End If, Select...Case
	Variables & Constants	Dim, As, New
	Procedures	PrivateSub, PublicSub, PrivateFunction, PublicFunction
Regular Expressions	PublicSub	"(?<=(?:\\n : ^)\\s*)(Public\\sSub.*)"
	PrivateSub	"(?<=(?:\\n : ^)\\s*)(Private\\sSub.*)"
	PublicFunction	"(?<=(?:\\n : ^)\\s*)(Public\\sFunction.*)"
	PrivateFunction	"(?<=(?:\\n : ^)\\s*)(Private\\sFunction.*)"
	DimAsNew	"Dim.*As.*New.*"
	AsNew	"Private.*As\\sNew.* Public.*As\\sNew.* Friend.*As\\sNew.*"
	ForEach	"For\\sEach.*"
	DoLoops	"Do\\sWhile.* Do\\sUntil.* Loop\\sWhile.* Loop\\sUntil.*"

Appendix D: The DUA's Code

Appendix D1: The generatedProxyStreamTCP.cs Client Proxy

```
[System.CodeDom.Compiler.GeneratedCodeAttribute("System.ServiceModel",
"3.0.0.0")]

[System.ServiceModel.ServiceContractAttribute(Namespace="http://AutoService_
_StreamTCP_MiniNet.ServiceModel.Agents",
ConfigurationName="IAutomation_StreamTCP_MiniNet")]
public interface IAutomation_StreamTCP_MiniNet
{
    [System.ServiceModel.OperationContractAttribute(Action="http://AutoService_
_StreamTCP_MiniNet.ServiceModel.Agents/IAutomation_StreamTCP_Mi" +
"niNet/RetRangeArrayList",
ReplyAction="http://AutoService_StreamTCP_MiniNet.ServiceModel.Agents/IAuto
mation_StreamTCP_Mi" + "niNet/RetRangeArrayListResponse")]
    string[] RetRangeArrayList();

    [System.ServiceModel.OperationContractAttribute(AsyncPattern=true,
Action="http://AutoService_StreamTCP_MiniNet.ServiceModel.Agents/IAutomatio
n_StreamTCP_Mi" + "niNet/RetRangeArrayList",
ReplyAction="http://AutoService_StreamTCP_MiniNet.ServiceModel.Agents/IAuto
mation_StreamTCP_Mi" + "niNet/RetRangeArrayListResponse")]
    System.IAsyncResult BeginRetRangeArrayList(System.AsyncCallback
callback, object asyncState);

    string[] EndRetRangeArrayList(System.IAsyncResult result);
}

[System.CodeDom.Compiler.GeneratedCodeAttribute("System.ServiceModel",
"3.0.0.0")]
public interface IAutomation_StreamTCP_MiniNetChannel :
IAutomation_StreamTCP_MiniNet, System.ServiceModel.IClientChannel
{
}

[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.CodeDom.Compiler.GeneratedCodeAttribute("System.ServiceModel",
"3.0.0.0")]
public partial class Automation_StreamTCP_MiniNetClient :
System.ServiceModel.ClientBase<IAutomation_StreamTCP_MiniNet>,
IAutomation_StreamTCP_MiniNet
{
    public Automation_StreamTCP_MiniNetClient()
    {
    }

    public Automation_StreamTCP_MiniNetClient(string
endpointConfigurationName) :
        base(endpointConfigurationName)
```

```

{
}

public Automation_StreamTCP_MiniNetClient(string
endpointConfigurationName, string remoteAddress) :
    base(endpointConfigurationName, remoteAddress)
{
}

public Automation_StreamTCP_MiniNetClient(string
endpointConfigurationName, System.ServiceModel.EndpointAddress
remoteAddress) :
    base(endpointConfigurationName, remoteAddress)
{
}

public
Automation_StreamTCP_MiniNetClient(System.ServiceModel.Channels.Binding
binding, System.ServiceModel.EndpointAddress remoteAddress) :
    base(binding, remoteAddress)
{
}

public string[] RetRangeArrayList()
{
    return base.Channel.RetRangeArrayList();
}

public System.IAsyncResult BeginRetRangeArrayList(System.AsyncCallback
callback, object asyncState)
{
    return base.Channel.BeginRetRangeArrayList(callback, asyncState);
}

public string[] EndRetRangeArrayList(System.IAsyncResult result)
{
    return base.Channel.EndRetRangeArrayList(result);
}
}

```

Appendix D2: App.config Configuration File

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <system.serviceModel>
    <bindings>
      <netTcpBinding>
        <binding name="NetTcpBinding_IAutomation_StreamTCP_MiniNet"
          closeTimeout="00:01:00" openTimeout="00:01:00"
          receiveTimeout="00:10:00" sendTimeout="00:01:00"
          transactionFlow="false" transferMode="Streamed"
          transactionProtocol="OleTransactions"
          hostNameComparisonMode="StrongWildcard" listenBacklog="10"
          maxBufferPoolSize="524288" maxBufferSize="65536"
          maxConnections="10" maxReceivedMessageSize="65536">
          <readerQuotas maxDepth="32"
            maxStringContentLength="8192" maxArrayLength="16384"
            maxBytesPerRead="4096" maxNameTableCharCount="16384"/>
          <reliableSession ordered="true"
            inactivityTimeout="00:10:00" enabled="false" />
          <security mode="Transport">
            <transport clientCredentialType="Windows"
              protectionLevel="EncryptAndSign" />
            <message clientCredentialType="Windows" />
          </security>
        </binding>
      </netTcpBinding>
    </bindings>
    <client>
      <endpoint address="net.tcp://j345-mm.uni.glam.ac.uk:8000/
        ServiceModelAgents/Service"
        binding="netTcpBinding"
        bindingConfiguration="NetTcpBinding_IAutomation_StreamTCP_MiniNet"
        contract="IAutomation_StreamTCP_MiniNet"
        name="NetTcpBinding_IAutomation_StreamTCP_MiniNet">
        <identity>
          <userPrincipalName value="mmhereeg@uni.glam.ac.uk" />
        </identity>
      </endpoint>
    </client>
  </system.serviceModel>
</configuration>
```

Appendix E: The Outcome of the Binning Process for Naïve Bayes Model

Fetch Size: 426	Refresh	Unbin	Filter	
Attribute Name	Value	Probability		
ASNEW COUNT	0	1.0000000000		
AVERAGE	0	0.1111111111		
AVERAGE	1	0.8888888889		
COUNT	2	0.9259259259		
COUNT	0	0.0740740741		
CPP FUNCTIONS	0	1.0000000000		
CUBE FUNCTIONS	0	1.0000000000		
DATABASE FUNCTIONS	0	1.0000000000		
DATE	1	1.0000000000		
DATE AND TIME FUNCTIONS	5	1.0000000000		
DAY	0	1.0000000000		
DO LOOPS COUNT	0	0.2592592593		
DO LOOPS COUNT	2	0.7407407407		
ENGINEERING FUNCTIONS	1	1.0000000000		
FINANCIAL FUNCTIONS	2	1.0000000000		
FOR EACH COUNT	0	1.0000000000		
INFORMATION FUNCTIONS	0	1.0000000000		
LOGICAL FUNCTIONS	1	0.8148148148		
LOGICAL FUNCTIONS	3	0.1851851852		
LOOKUP AND REFERENCE FUNCTIONS	3	0.4444444444		
LOOKUP AND REFERENCE FUNCTIONS	2	0.3703703704		
LOOKUP AND REFERENCE FUNCTIONS	5	0.0370370370		
LOOKUP AND REFERENCE FUNCTIONS	4	0.1481481481		
MATHEMATICAL FUNCTIONS	1	1.0000000000		
MAX	0	0.0370370370		
MAX	1	0.9629629630		
MIN	1	0.8148148148		
MIN	0	0.1851851852		
MONTH	0	1.0000000000		
NUMBER OF CHARTS	0	1.0000000000		
NUMBER OF PIVOT TABLES	0	1.0000000000		
PRIVATE FUNCTION COUNT	0	0.2592592593		
PRIVATE FUNCTION COUNT	2	0.7407407407		
PRIVATE SUB COUNT	0	0.2592592593		
PRIVATE SUB COUNT	1	0.7407407407		
PUBLIC FUNCTION COUNT	0	1.0000000000		
PUBLIC SUB COUNT	0	1.0000000000		
STATISTICAL FUNCTIONS	1	1.0000000000		
SUM	2	0.9629629630		
SUM	6	0.0370370370		
TEXT FUNCTIONS	2	1.0000000000		
TIME	0	1.0000000000		

Appendix F: The Naïve Bayes Apply Activity Results

DMR\$CASE_ID	PREDICTION	PROBABILITY	COST	RANK	AUTHOR_ID
8,171,459	1	1	0	1	1
9,003,800	1	1	1	1	2
9,047,891	1	1	0	1	3
9,133,623	1	1	0	1	5
9,161,015	1	1	0	1	6
9,165,185	1	1	0	1	7
9,161,074	1	1	0	1	8
7,227,140	1	1	0	1	10
8,038,945	1	1	0	1	11
8,010,846	1	1	0	1	12
8,046,425	1	1	0	1	15
9,001,441	1	1	0	1	17
9,001,549	1	1	0	1	18
9,001,557	1	1	0	1	19
9,001,670	1	1	0	1	20
9,001,867	1	1	0	1	21
9,002,405	1	1	0	1	22
9,002,413	1	1	0	1	23
9,002,510	1	1	0	1	24
9,002,758	1	1	0	1	25
206160560	2	1	0	1	99
208,038,944	2	1	0	1	100
209,110,512	2	1	0	1	180
209,116,496	2	1	0	1	181
209,116,736	2	1	0	1	182
209,117,984	2	1	0	1	183
209,118,000	2	1	0	1	184

209,119,296	2	1	0	1	185
209,120,800	2	1	0	1	186
209,126,320	2	1	0	1	187
209,127,408	2	1	0	1	188
209,136,128	2	1	0	1	189
209,160,928	2	1	0	1	190
209,161,008	2	1	0	1	191
209,161,072	2	1	0	1	192
209,165,184	2	1	0	1	193
209,060,480	2	1	0	1	194
209,059,200	2	1	0	1	195
209,052,848	2	1	0	1	196
209,006,096	2	1	0	1	197
406,160,576	3	1	0	1	198
408,010,848	3	1	0	1	199
408,038,944	3	1	0	1	200
408,050,240	3	1	0	1	204
408,244,416	3	1	0	1	205
409,001,408	3	1	0	1	206
409,001,440	3	1	0	1	207
409,001,536	3	1	0	1	208
409,001,568	3	1	0	1	209
409,001,664	3	1	0	1	210
409,001,856	3	1	0	1	211
409,002,400	3	1	0	1	212
409,002,400	3	1	0	1	213
409,002,496	3	1	0	1	214
409,002,752	3	1	0	1	215
409,002,816	3	1	0	1	216
409,003,072	3	1	0	1	217

409,003,136	3	1	0	1	218
409,003,136	3	1	0	1	219
409,003,520	3	1	0	1	220
509,049,280	4	0.9998	0.0011	1	426
509,049,568	4	0.982	0.1258	1	427
509,052,864	4	0.982	0.1258	1	428
509,053,600	4	0.9807	0.1346	1	429
509,055,520	4	0.982	0.1258	1	430
509,058,336	4	0.9807	0.1346	1	431
509,059,200	4	0.9998	0.0011	1	432
509,116,480	4	1	0	1	433
509,116,736	4	0.9607	0.2742	1	434
509,117,696	4	0.9732	0.187	1	435
509,117,984	4	0.982	0.1258	1	436
509,120,800	5	0.5962	3.5107	1	437
509,126,304	4	0.9607	0.2742	1	438
509,127,392	4	0.982	0.1258	1	439
509,133,632	4	0.982	0.1258	1	440
509,136,128	4	0.982	0.1258	1	441
509,160,928	4	0.9384	0.4304	1	442
509,161,024	4	0.982	0.1258	1	443
509,161,088	4	0.982	0.1258	1	444
509,165,184	4	0.982	0.1258	1	445
609,049,600	5	1	0	1	507
609,053,630	5	0.9997	0.003	1	508
609,055,550	5	0.999	0.0089	1	509
609,058,370	5	1	0	1	510
609,058,370	5	0.9999	0.0005	1	511
609,059,200	4	0.982	0.1258	1	512
609,060,480	5	1	0	1	513

609,110,530	5	0.9999	0.0011	1	514
609,116,480	5	0.9943	0.0496	1	515
609,116,740	5	1	0	1	516
609,117,700	5	0.9651	0.3032	1	517
609,118,020	5	0.9999	0.0005	1	518
609,119,300	5	1	0	1	519
609,120,830	5	1	0	1	520
609,126,340	5	1	0	1	521
609,127,420	5	0.9999	0.0011	1	522
609,165,180	5	1	0	1	523
609,136,130	5	1	0	1	524
609,160,900	5	0.9943	0.0496	1	525
609,161,020	5	1	0	1	526