World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:6, No:11, 2012

The Development of the Multi-Agent Classification System (MACS) in Compliance with FIPA Specifications

Mohamed R. Mhereeg

***Abstract*—**The paper investigates the feasibility of constructing a software multi-agent based monitoring and classification system and utilizing it to provide an automated and accurate classification of end users developing applications in the spreadsheet domain. The agents function autonomously to provide continuous and periodic monitoring of excels spreadsheet workbooks. Resulting in, the development of the MultiAgent classification System (MACS) that is in compliance with the specifications of the Foundation for Intelligent Physical Agents (FIPA). However, different technologies have been brought together to build MACS. The strength of the system is the integration of the agent technology with the FIPA specifications together with other technologies that are Windows Communication Foundation (WCF) services, Service Oriented Architecture (SOA), and Oracle Data Mining (ODM). The Microsoft's .NET widows service based agents were utilized to develop the monitoring agents of MACS, the .NET WCF services together with SOA approach allowed the distribution and communication between agents over the WWW that is in order to satisfy the monitoring and classification of the multiple developer aspect. ODM was used to automate the classification phase of MACS.

***Keywords*—**Autonomous, Classification, MACS, Multi-Agent, SOA, WCF.

I. Introduction

VARIOUS methods for classification of End Users Computing (EUC) activities have been proposed in the literature [2] [3] [8] [19] [25]. 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 paper 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 monitoring and data gathering process that leads to the classification of end user developers. The Monitoring Agents (MA) were configured to execute automatically, without any user intervention as windows service processes in the .NET web server application of the system. Additionally, these agents function autonomously to provide continuous and periodic monitoring of excel spreadsheet workbooks. These agents listen to and read the contents of workbooks 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 will be collected and filtered by a Database Updater Agent (DUA) 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. The .NET WCF services were used for the distribution of the agents over a network to satisfy the monitoring and classification of the multiple users approach. Spreadsheet applications will be used as end users workbench in order to evaluate the system, the reason is that 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 [23]. 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.

Thus MACS 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.

II. The Agent Based Windows Service

An agent based on the concept of a .NET Windows Service has been explored by [24][10][11]. Windows services enable developers to create long-running executable applications that run in their own Windows sessions [12]. 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 [12]. This concept has been used in the construction of the data gathering and classification system that is configured to start

M. R. Mhereeg was with Glamorgan University, Pontypridd, CF37 1DL, UK. He is now with the Department of Data Analysis and Computing, Aljabal Algharbi University, Gharian, Libya (e-mail: mmhereeg@msn.com).

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.

III. Data Collection

The agent based monitoring system operates autonomously as service running in its own thread space, continuously watching for files being saved to disk. The agent is then tasked with identifying from all the disk activity spreadsheet workbooks. The saved spreadsheets will then be analysed in terms of: File and Author Properties, and Number and Type of Functions used (see table 1), in addition to the add-ins features and charts. The 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 a program environment include the number and type of: selection, sequence, iteration, procedures, functions and variables and constants.

For the scope of this paper, Excel spreadsheets have been used as a testing environment because it offers functionality and a variety of features explained above that could be used to express the different levels of users' development and operation. A typical user may start off as a simple consumer of data and progress through simple charts, formulas, automating tasks with VBA macros eventually leading to publishing data on the web. These features can be programmatically and automatically monitored and analyzed via windows service based agent to measure complexity of spreadsheets and classify end users.

TABLE I
Software Usage and Object & Code Patterns

Number of Functions	**Number of Iterations**
Database	Do … Loop Until
Lookup & Reference	Do … Until Loop
Date & Time	Do… Loop While
Math & Triggers	Do …While Loop
Financial	For … Next
Information	For … Each
Logical	**Number of Selections**
Text	If … Then … End If
Cube functions	If … Then … Else … End If
Engineering functions	Select … Case
Add-ins and automaton	**File Properties**
Functions	
Calculate number of variables and constants	Author
Constants	Company
Variables	Version
Calculate number of Procedures	Date Last Saved
Function	Date Created
Sub	Comments
Applications & Languages	Document Name
Automated applications	Path Name
Windows API calls	Application
ADO	Last Saved By
C/C++	Title
VB6 or VB.net	Subject
XML	

IV. Overview of FIPA Architecture

In 2002 FIPA published the standards describing the multi-agent systems [5][6]. 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 Fig. 1). 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.

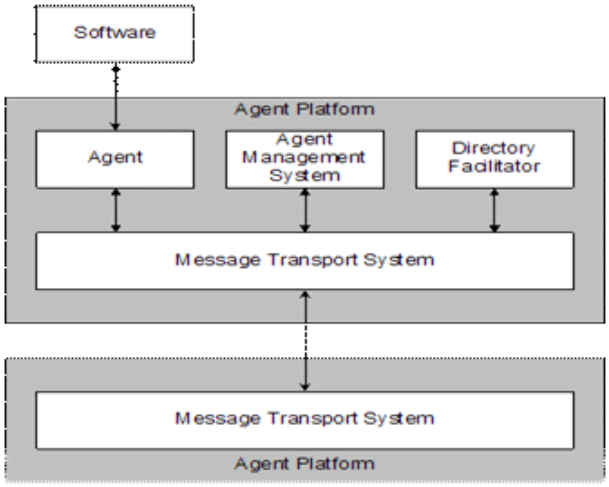

Fig. 1 Agent Management Reference Model

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.

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 [7]).

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.

V. 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 [22][28]. 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) [9][15]. The Service Oriented Architecture paradigm is presented in Fig. 2.

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

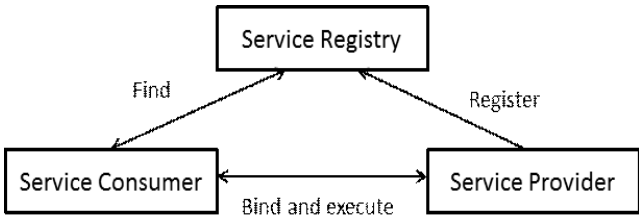

Fig. 2 Service Oriented Architecture Schema

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, 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 [9][15].

Fig. 3 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.

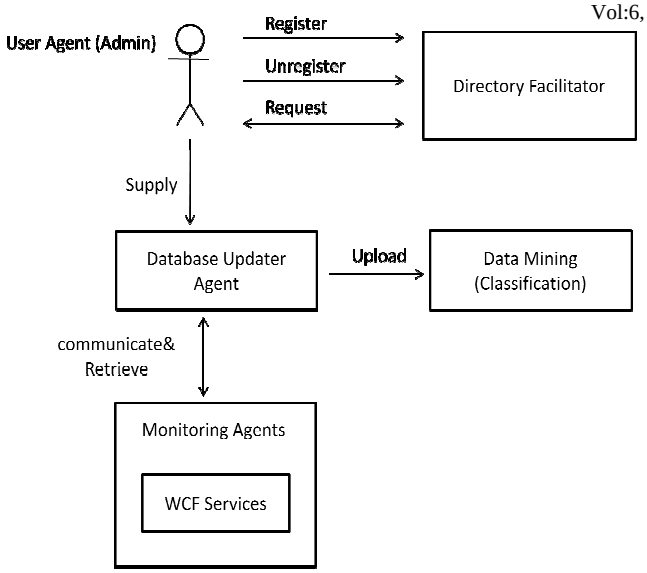

Fig. 3 Architecture skeleton of the MACS

VI. MACS Multi- Agent System

Using the techniques mentioned in the previous sections, 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 parallely.

- 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).

Fig. 4 represents the MACS multi-agent based architecture and the interaction between the agents.

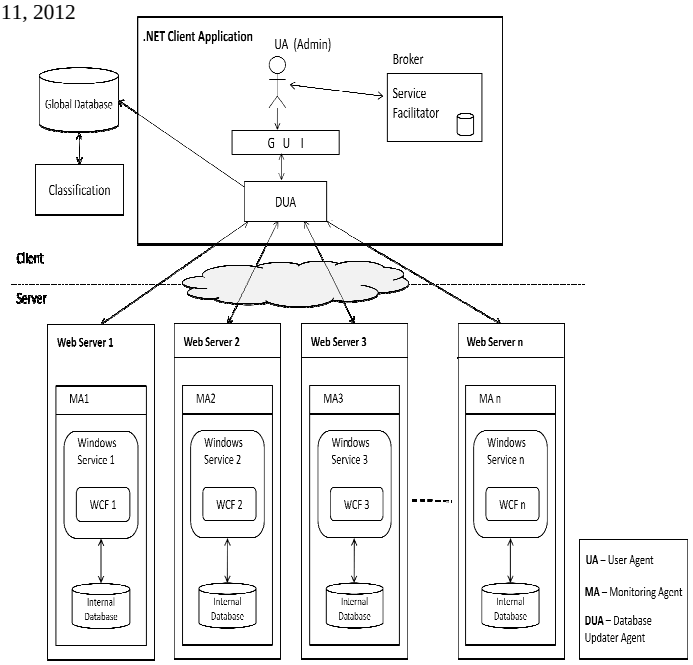

Fig. 4 MACS Architecture

- 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 [5]. 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 [29]. 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 [5]. It is used to store the agent details like the agent names and addresses in order to be used for the communication process [5][6]. 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.

VII. 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.

VIII. 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 (SF) 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.

IX. Software Development Methodology 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 [20]. Numerous agent oriented methodologies have been proposed in the literature such as Prometheus [16][17], GAIA [30], TROPOS [21][4] and MaSE [26][27], 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 why Prometheus has been chosen for the design of the agents.

A. 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 [18]:

- 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, 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.

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 [13][14]. The features of Prometheus distinguish it from these methodologies, but none of them have all of the Prometheus features described above [18].

X. Excel Developer Categories

Excel developers can be allocated to five different categories [1]. 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 II adapted from [1] summarizes the categories. Table II summarizes the categories together with a description and the level of knowledge (Usage) required for every category.

TABLE II
Excel Developer Categories

Category	Description
Excel User	- Store lists, Simple repetitive calculations - Some worksheet functions - Pivot tables - Charts
Excel Power User	- Wide understanding of excel Functionality - 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	- 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	- 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	- 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 For Example: - 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.

XI. Conclusion

This paper presented 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. This will assist management to accurately allow for precise tailor training activities for future spreadsheet development. This approach focused on the task of developing 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. However, 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. It has also been demonstrated that the .NET WCF services technology can be utilized 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. MACS has been built based on SOA, this supplied the system with more flexibility and the interaction between the agents is simplified. Although MACS has been developed as a final product, this paper presented only the architecture of it, the design, implementation, evaluation and results will be followed in the papers to come.

REFERENCES

[1] 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.
[2] Codasyl end-user facilities committee status report. Information Management Two,North Holland. 1979, 137-163.
[3] Cotterman, W. & Kummar, K. (1989) User Cube: A Taxonomy of End Users. *Communication of the ACM,* 32**,** 1313-1320.
[4] 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.*
[5] FIPA (2002) FIPA Abstract Architecture Specification.
[6] FIPA (2004) FIPA Agent Management Specification.
[7] FIPA Agent Message Transport Service Specification, http://www.fipa.org/specs/fipa00067/, (last visited 2010).
[8] Govindarajulu, C. (2003) End Users: Who are They? *Communication of the ACM,* 46**,** 152-159.
[9] Hasan, J. & Duran, M. (2006) *Expert service-oriented architecture in C# 2005*, Apress.
[10] 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.
[11] 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 elecommunications and Computer Networks.* Portsmouth, United Kingdom.
[12] Introduction to Windows Service Applications, MSDN, http://msdn.microsoft.com/en-us/library/d56de412(VS.80).aspx, (last visited, 2010).
[13] 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.
[14] 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).
[15] Krafzig, D., Banke, K. & Slama, D. (2005) *Enterprise SOA: Service-oriented Architecture Best Practices*, Englewood Cliffs, Prentice Hall PTR.
[16] LIN, P. & Michael, W. (2002) Prometheus: A methodology for developing intelligent agents. *Third International Workshop on Agent-Oriented Software Engineering.*
[17] LIN, P. & Michael, W. (2002) Prometheus: A pragmatic methodology for engineering intelligent agents. *In Proceedings of the OOPSLA 2002 Workshop on Agent-Oriented Methodologies.*
[18] LIN, P. & Michael, W. (2004) The Prometheus Methodology.
[19] Mclean, E. R. (1979) End-users as application developers. *MIS Quarterly,* 10(4)**,** 37-46.
[20] Michael, L., Peter, M. & Chris, P. (2003) Agent technology: Enabling next generation computing: A roadmap for agent-based computing. AgentLink report, vailable from www.agentlink.org/roadmap, ISBN 0854 327886.
[21] 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.
[22] Peiris, C., Mulder, D., Cicoria, S., Bahree, A. & Pathak, N. (2007) *Pro WCF: practical Microsoft SOA implementation*, Apress.
[23] Price Waterhouse and Coopers. (2004) The Use of Spreadsheets: Considerations for Section 404 of the Sarbanes-Oxley Act. Price, Waterhouse, Coopers.
[24] REA, S. (2005) *Building Intelligent .NET Applications,* Chichester, Great Britain, Wiley.
[25] Rockart, J. F. & Flannery, L. S. (1983) The management of end-user computing. *Commun. ACM,* 26(10)**,** 776-764.
[26] Scott, A. & Deloach (2001) Analysis and design using MaSE and agent Tool. *In Proceedings of the 12th Midwest Articial Intelligence and Cognitive Science Conference (MAICS 2001).*
[27] Scott, D., Mark, W. & Clint, S. (2001) Multiagent systems engineering, *International Journal of Software Engineering and Knowledge Engineering,* 11(3)**,** 231-258.
[28] Vasters, C. (2005) Introduction to Building WCF Services , http://msdn.microsoft.com/en-us/library/aa480190.aspx, MSDN (last visited 2010).
[29] Windows Communication Foundation Reliable Sessions, http://msdn.microsoft.com/en-us/library/ms733136.aspx, MSDN, (Last Visited 2010).
[30] 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).