https://dx.doi.org/10.22161/ijaers/3.12.1 *ISSN: 2349-6495(P) | 2456-1908(O)*

A Communication System Architecture Based on Sharing Information to Integrate Components of Distributed Multi-Agent Systems within an IT GRC Platform

S. Elhasnaoui, L. Moussaid, H. Medromi, A. Sayouti

(EAS- LRI) Systems Architecture Team, ENSEM, Hassan II University, BP.8118, Oasis Casablanca, Morocco

Abstract— *To build complete and complex distributed platform, researchers are used to applying standards that facilitate sharing information as well as distributed service-oriented architectures that provide reuse and interoperability by means of component integration. These concepts lead us to a communication system that will allow us to implement tools that give full support to the management of IT GRC processes, taking advantage of the synergy effect created by the integration of the different components.*

Thus, throughout this article we analyze the proposals from the most relevant consortia concerned with communication system standards. This analysis will demonstrate that the use of standardized middleware technologies can limit the reuse and interoperability. Then, we will show a proposal which tries to solve this shortfall, using a blackboard-based architecture for integrating and communicating heterogeneous distributed components within an IT GRC platform.

Keywords— Communication, architecture, IT GRC, Tuple space, Web service, Agent, blackboard.

I. INTRODUCTION

To support the construction of distributed systems, we are witnessing a changing of the models that make intensive use of software engineering, analysis models, etc., to facilitate the implementation of these systems. However, implementation of distributed systems is not linked to a specific communication system that manages workflow between these applications. Nevertheless, the technology choice of the most appropriate communication system is a fundamental task for ensuring the integration of components and system scalability.

Our team of Corporate Governance of ICT which belongs to LISER laboratory of the University Hassan II Casablanca has developed a platform to support good governance, risk management and compliance of information technology and communication within an enterprise, including a plurality of distributed systems that:

- Ensures and assess intelligently align enterprise business goals with the goals and IT strategy,
- Manages IT processes
- Prioritizes IT investments in line with the contribution of business value.
- Manage IT risk and evaluate,
- Ensures compliance with the legal framework,
- Choose the best repository of Governance, Risk and Compliance Information Systems to perform the tasks mentioned above.
- Update repositories according to the latest existing versions on the market,

The platform is based on the standards and methods of Governance, Risk and Compliance Information Systems (IT GRC), the most common (COBIT, ITIL, PMBOK, ISO27001, ISO27002, ISO27005, Mehari, EBIOS) This is a distributed platform based on multi-agent systems consist of a strategic layer, layer and layer decision processing. However, it is necessary to develop a communication system that enables workflow management within the IT GRC platform. Therefore, not only allowing the integration of distributed systems, but also agents and components that support process management Governance, risk and compliance through IT GRC platform, and to do from disparate treatment of different sources, are additional goals that we have set as a challenge in our research. The communication system that manages workflow between the components of the IT GRC platform will manage IT GRC during its various stages, to build an adaptive and intelligent platform based on components and shared resources.

There are several technological proposals for managing the communication for the construction of heterogeneous distributed environments such as Web services, CORBA, ICE, Java RMI, .NET, etc., some of which will be analyzed in the next paragraph. Typically, these technologies are based on providing an environment in which we can save the system and launch the objects that implement them. The communication system is responsible for providing a reference to the object or

https://dx.doi.org/10.22161/ijaers/3.12.1 *ISSN: 2349-6495(P) | 2456-1908(O)*

objects that implement the functionality associated with a particular system. However, to have access to them, it is necessary to know their public interfaces. In this sense, it is difficult to add systems that generate added value if they are not pre-defined and standardized. Some technologies that manage workflows will be analyzed in the next paragraph.

II. STATE OF THE ART: COMMUNICATIONS SYSTEM TECHNOLOGIES FOR THE IMPLEMENTATION OF DISTRIBUTED HETEROGENEOUS ARCHITECTURES

As noted above, because of their nature-oriented service, reference architectures tend to provide implementations based on Web Services for the implementation of applications between different layers of a distributed system (WS) (http://www.Webservices.org/). WS allows encapsulating business logic components regardless of programming language and platform. It is based on HTTP and uses XML (most times) to encode the requests of services following the recommended protocol SOAP (Simple Object Access Protocol) of the W3C. However, WS is not a communication system for the implementation of distributed systems, but for the specification of services whose interface is described using WSDL (Web Service Definition Language).

This feature provided by the WS might also be obtained using CORBA (Common Object Request Broker Architecture) (http://www.corba.org/). In this aspect, it is necessary to mention the effort of the "CORBAlearn" (Anido and Llamas 2001a ;. Anido et al, 2001b), the Ibero-American Network Telematic RICOTEL where a domain CORBA is intended to provide standardization in the interface definition learning services online using sound definition language interface definition (IDL). Among the advantages of CORBA, he built a compact binary transport protocol directly on top of TCP / IP, which has a direct impact on performance (Demarey et al., 2005). While CORBA can resolve WS deficiencies, the facts indicate that the WS is the alternative most sustained for most of the industry. In this sense, Baker (2002) offers a solution that manages the integration of distributed systems, so there is not a universal solution, but there are solutions to every problem and building bridges between management technologies communication flows, all solutions can be integrated and interoperable.

On the other hand, Henning zeroc (http://www.zeroc.com/ice.html) tries to give an explanation to support WS and marginalization of CORBA (Henning, 2006a), the analysis of the social situation , economic and technological factors that made CORBA technology relegated to the niche of real time and embedded systems. Thus, Henning (2006a) says, "CORBA was a victim of trends and fashion industry." Similarly, Henning stressed the technical complexity of CORBA, its lack of certain features such as security and support for the control, the difficulty of building a good distribution service, lack of asynchronous method invocation / dispatching among server customers, and the lack of mapping languages like C # and Visual Basic, all these factors put CORBA outside the .NET architecture.

To resolve this situation, zeroc developed ICE (Internet Communication Engine). Its authors tried to "build a middleware platform that is as powerful as CORBA, without making any mistakes CORBA" (Henning and Spruiell, 2006b). With this solution, ICE is an object oriented middleware for heterogeneous distributed environments. In addition, it provides a set of features such as security, distribution of the event and support for asynchronous invocation / dispatching method, among others.

We can find other middleware technologies that are more oriented towards specific programming platforms and languages. Examples: Java RMI (Remote Method Invocation) and the Microsoft .NET platform. The first uses Java to achieve cross-platform services. However, it is a middleware in which Java is the single programming language for the deployment of services. Furthermore, the .NET platform allows building heterogeneous distributed systems using different programming languages such as C # or Visual Basic .NET, but both the programming language and communication protocols should be those supported by the platform.

Table :1.Summarizes the characteristics of the cited technologies for managing communication between distributed systems.

Comparison of some middleware technologies.

Feature	Web services	CORBA	ICE	Java RMI	.Net
Services interface specification	Yes (WSDL)	Yes (IDL)	Yes (Slice)	Yes (Java)	Yes (.Net related languages)
Platform independent	Yes	Yes	Yes	Yes (Through JavaVM)	No
Multi-language support	Yes	Yes	Yes	No	Yes (but only for .Net related languages)

Until now all these middleware technologiesare indicated according to a well-knownand standardized definition of the interfaceservices. The considerationis inother types ofmiddlewaretechnologiesthat allow the incorporationof

https://dx.doi.org/10.22161/ijaers/3.12.1 *ISSN: 2349-6495(P) | 2456-1908(O)*

agents andintelligent services. In this paper,we willpropose architectureof communication thatis based onthe sharing of informationascommunication modewhichcombines the technologyof multi-agentsystemsand web servicesto manageworkflowsbetween the componentsof the platformIT GRC.

Thecommunication architecturesusing sharinginformation mode:

They are known as middleware based on tuple spaces (Gelernter, 1985), in which a shared associated memory is used as a means of communication between different agents, services and integrated systems. Thus, they represent an alternative to the remote method invocation imposed (and limited) by a public interface. Distributed systems, agents and services can write and read information from a shared memory accessible by the network.

Moreover, in more general terms, choose either a middleware for the implementation of a distributed system, depend on the characteristics to consider: programming languages, performance and bandwidth, the architectural decisions such as topology and scalability, services to be provided by the middleware for the deployment and maintenance of applications, platforms that can be used by the various objects that make up the system, etc.

Most middleware mentioned so far takes as its starting point a definition of systems that will compose a distributed platform. This principle facilitates the implementation of environments of different paradigms of IT GRC (strategic management, setting the company making the decision, processing of IT processes ...).

However, with regard to our work, we modeled a communication system to build distributed IT GRC platform consists of heterogeneous systems to enable us to implement them effectively and efficiently. Moreover, this integration should allow the incorporation of intelligent agents and web services to support the management of IT GRC.

Among the architectures used to implement a communication system we find that based on information sharing (or blackboard architecture). This architecture is based on the metaphor "blackboard". The main idea is to build and add new information in a spirit of cooperation, from the information that is available on a shared basis that all participants can see.

The communication architecture for sharing information like this: a group of people are placed in front of a blackboard where the definition of a problem is written. Then someone starts writing his contribution. Thereafter, the one who has something to add will take a turn and add that information on the blackboard. Then someone else will take a turn and read the information generated by the previous person so they can contribute to the array. This will be repeated until the solution is reached. In other words, the table is the common base of knowledge, from the specification of the problem until the culmination of the solution of the problem it is updated iteratively.

To achieve the implementation of the system, this metaphor seeks to benefit from the synergy created by the integration of different agents to become a party. Officers use the information available on the blackboard to generate new information that can be used by other agents with specific functionality.

This metaphor, which is the basis of the blackboard architecture, is used by systems based on tuple spaces. These systems have been introduced with the Linda coordination language (Gelernter, 1985). A tuple space is a shared memory, where information is organized as a set of tuples. A tuple is the key element of the system. It consists of a vector field containing the data type values. In such systems, the producers Agents send data to the tuple space (they write on the blackboard) and consumers officers receive data from the tuple space, if they match a certain pattern (see the table that information they can understand or can their interest).

The use of tuple spaces to the modeling of a communication system within a distributed platform has several advantages that Krummenacher et al. (2005) point out, namely:

• there is not a need to address the communication participants directly;

• there is no need for synchronous links between communication participants;

• there is no need to run in the same environment as the access to the same space is guaranteed.

In addition, Krummenacher et al. (2005) state that "Decoupling has obvious advantages modeling to define reusability, distribution, heterogeneity and agility communications solutions”

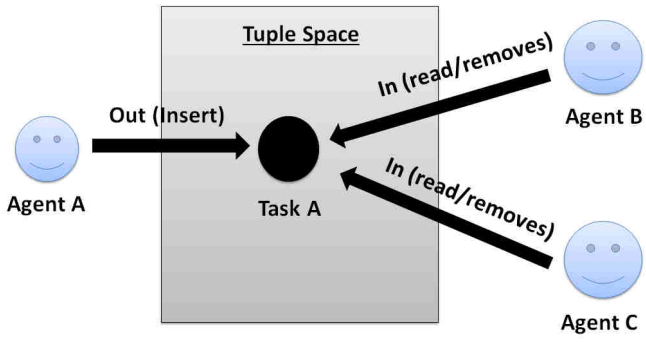

Fig.1: An abstract representation of the schedule tuple space

Similarly,an interesting featurein the communication betweenthe different agentsthat interactthrough thetuplespace is theevent notification.With this,the server sends anotificationto the systemsinterested in receivinga specific eventwhena tuplecorresponding to aspecific modelhas been written,updated ortaken. This meansthat

https://dx.doi.org/10.22161/ijaers/3.12.1 *ISSN: 2349-6495(P) | 2456-1908(O)*

the server isnot onlya tuple ofdeposit, but it is alsoan activepartof the system thatallows building event environments.

III. OVERVIEW OF IT GRC PLATFORM

Before presenting the detailed communication system architecture, we should locate it within the global IT GRC platform. This latter solution aims to combine IT GRC frameworks in an intelligent manner to take the right IT decision for business directives.

It focuses on business objectives and proposes the best solution for an efficient IT management.

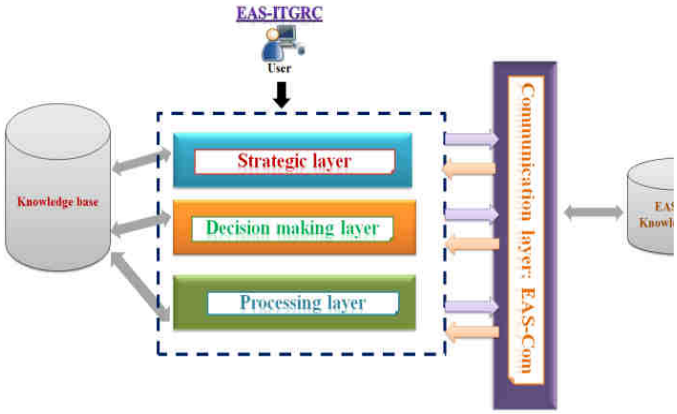

Fig.2: IT GRC solution: Proposed architecture

IT GRC platform is composed of four principle layers:

- **Strategic layer:** based on COBIT framework; ensuring permanent alignment of IT and Business with stakeholders' participation. The output of this layer is the proposition of IT processes that must be managed.
- **Decision layer:** assures the choice of the best framework for each IT process and adds a decision-making aspect on every platform of the layer processing
- **Processing layer**: This layer is composed of different systems, which can be implemented; each of these systemsrelies onapreciseIT framework formanagingITprocesses defined bythe strategylayer
- **Communication layer**: It is responsible for all communications between layers of the IT GRC platform. It is a distributed system which is composed of three sub system that ensures all communication between strategic, decision and processing layers

IV. EAS-COM:COMMUNICATION SYSTEM ARCHITECTURE BASED ON MULTI AGENTS SYSTEM USING SHARING INFORMATION MODE

Communication System:The architecture ofEAS-COM is based oninformation sharing. For this,we choseatuplespaceserver, so that the different entities(agents andservices)will interacttogether usingtuples.In addition,thissystem will providestorage requirementsthatthe system may have. Among theinformation stored in theserver, we can find theusersession data communication messages used in different tools.

After outlining the architectureof the communication systemand identifiedthe necessary parts, we can take a look atsome partsof it. Solet's get intothe detailsabouteach service andeachofficer, explaining their specificfeatures andimplementationsissues.

The overall architecture accounts for three layers, each responsible of a different aspect of the system (see Fig. 3): the interaction layer, responsible for collecting web services from/to IT GRC platform's applications (EAS-Strategic, EAS-Decision, EAS-processing) that EAS-COM engine orchestrates; the correlation layer, interpreting messages and their correlation sets, and then routing messages to the proper process instance; and the workflow layer, executing the specific workflow activities that concern each process instance.

https://dx.doi.org/10.22161/ijaers/3.12.1 *ISSN: 2349-6495(P) | 2456-1908(O)*

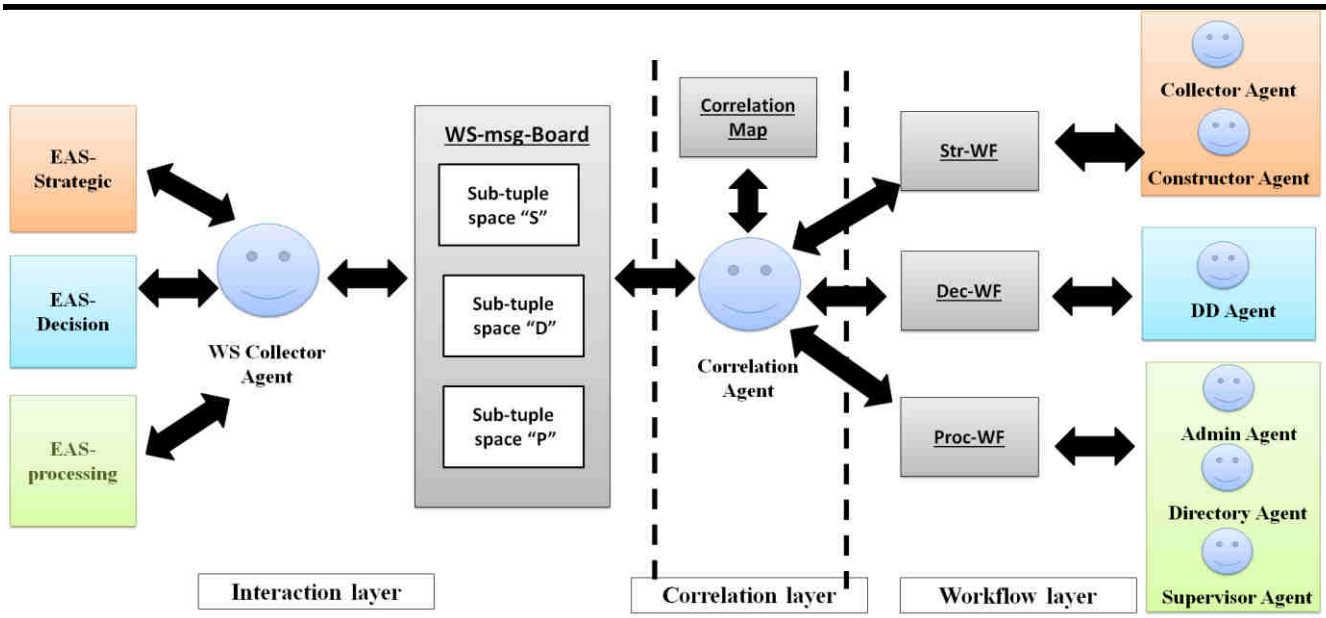

Fig.3: Architecture of the communication system. Based on multi-agent system and web service and using tuple spaces for sharing information between IT GRC platform components.

The interaction Layer

In the interaction layer, a tuple space called ws-msg-board is first used to collect all the incoming/outgoing request messages, encoded in tuples of the kind:

Ws-msg-request(ID,user,date,Body)

A bridge agent called WS-Collector-Agent is used to receive WS requested (IT Services) from EAS-Strategic and correspondingly insert them into the **Sub-tuple space "S"** as soon as they arrive. After, the WS-Collector-Agent collects the ws-msg result (IT Service categorized) as soon as they are inserted in the **Sub-tuple space "D"** by the agents in the workflow layer, then prepares the REST message and sends it to the EAS-Decision.

WS-Collector-Agent is used to receive WS requested (IT Service Decided) from EAS-Decision and correspondingly insert them into the **Sub-tuple space "D"** as soon as they arrive.

Next, the WS-Collector-Agent collects the ws-msg result (IT Service Processed) as soon as they are inserted in the **Sub-tuple space "P"** by the agents in the workflow layer, then prepares the REST message and sends it to the EAS-Processing.

WS-Collector-Agent is used to receive WS result (IT Service Report) from EAS-Processing and correspondingly insert them into the **Sub-tuple space "P"** as soon as they arrive.

All of WS-msg must maintain the same ID which is the request identifier as above.

The correlation layer

In the correlation layer, an agent called Correlator Agent is in charge of creating process instances, correlating incoming messages to them, and accordingly performing message routing. In particular, a tuple Space called correlation map is used by Correlator Agent to register information related to the created and currently-running process instances.

The Correlator Agent first retrieves ws-msg-request from the ws-msg-board tuple space. Then, depending on their bodies' structure, it accesses the values as written in the message, and checks whether an existing tuple space instance of processing activities in correlation map has equal values. Else, if no instance of processing activities exists a new processing activity is created in the workflow layer and the message is routed to it.

The workflow layer:

The workflow layer is the part of the engine that carries on the workflow of each process instance. We describe first the main architecture of the agent-based system which handles each different process instance, and then how messages exchange is supported.

- **Strategic-COM**

In the workflow layer, each process instance is executed by Strategic-COM agents sharing the tuple space **Str-WF**:

Manager Agent: retrieve WS-msg-request instance from tuple space **Str-WF** (created by correlator Agent) and categorize the IT processes (belonging to the received web service) according to frameworks of best practices of the three disciplines of IT GRC (IT governance: ITIL, ISO/IEC 27002, PMBOK, CMMI…, IT Risk: EBIOS, MEHARI, ISO/IEC 27005…, IT Compliance: SOX or 08:09 laws). Next, Manager Agent registers the result of categorization into the same tuple space

IT associates each IT processes belonging to a discipline of IT GRC and later to one or more IT frameworks.

Matrix building Agent: it accesses the values as written in Str-WF by Manager Agent, and checks whether all IT

https://dx.doi.org/10.22161/ijaers/3.12.1 *ISSN: 2349-6495(P) | 2456-1908(O)*

Processes of Service IT requested are categorized, retrieve the result of categorization and written in the same tuple space the matrix that has formed as following:

$$[\{proc1, \begin{matrix}Fram\ i\\Fram\ j\end{matrix}\}, \{Proc2 \begin{matrix}Fram\ a\\Fram\ b\end{matrix} \ldots.\}]$$

Once the categorization is accomplished, the Correlator agent matches it with adequate IT Service through its ID, and inserts the service IT categorized into Correlation map tuple space. After that, it translates it as WS-msg to be introduced into Sub-TupleSpace D.

- **Decision-COM**

WS collector Agentaccesses to Sub-TupleSpace D andsend theweb servicethat supportsITservice categorized toEAS-Decision. This last one replies to this request and send ws-msg-request that contains IT Service Decided. WS collector Agent collects this web service and inserts it into Sub-TupleSpace D. After, the correlation Agent retrieves it and stores it into correlation map tuple space.
It checks the web service structure and inserts into tuple space **Dec-WF.**

Agent DP: retrieve WS (the IT Service Decided) instance from tuple space **Dec-WF** (created by correlator Agent), verifies its syntax and inserts it into the same tuple space. Next, the correlation Agent introduces it into **Proc-WF.**

- **Processing-COM**

Agent Admin: accesses to **Proc-WF** tuple spaceandchoose processing system that will handle each IT process the IT service Decided. It manages the list of processing systems that exist in its knowledge base. The list contains information of all available processing systems: The name and description of the system, framework required, and the IP address of the computer where the processing system is running. However, there is dynamic information stored in this list that is constantly modified: the system performance, the number of execution and the quality of system. All this information istaken into consideration bythis agent todecide whichtendering systemmay more preferablyto manage the process.
Once the destination of processing system is done, the Correlator agent matches each IT process with its system processing and with adequate IT Service through its ID, and inserts these requests (IT Process 1, Sys-Proc i, Process 2, Sys-Proc j, Process 3, Sys-Proc k…) into Correlation map tuple space. After that, it translates it as sub-WS-msg to be introduced into Sub-TupleSpace P.
WS collector Agentaccesses to Sub-TupleSpace P andsend the sub web servicesthat supportsITservice processing to one or many EAS-Processings according to number of processing demands (system processing chosen). These last one reply to these requests and each one reply by sending WS that contains reports of treatment. WS collector Agent collects this web service and inserts it into Sub-TupleSpace P. After, the correlation Agent retrieves them and stores them into correlation map tuple space. It checks the web services structure and inserts into tuple space **Proc-WF.**

Directory Agent: accesses to **Proc-WF** tuple space and retrieve these processing reports (associated with their IT service parameters) in order to verify their structure and store them into database.

Supervisor agent: accesses to **Proc-WF** tuple space and retrieve these processing reports (associated with their IT service parameters) in order to calculate the performance of each processing system and increment its number of execution. These informations will be registered in the database to be taken into consideration by Admin agent to choosethe best system forthenextqueries.

V. CONCLUSION

Designing and developing a communication system is a hard task because of the complexity of managing interaction and concurrency in a general way. Sharing information mode of communication allow such complexity to be faced at an adequate abstraction level, by providing constructs and mechanisms explicitly designed for manipulating interactions and shaping coordination flows.
Thus, we have proposed as a solution the tuple spaces middleware technologies that use blackboard architecture. This will allow ad hoc service integration by exchanging information in a centralized shared memory. We have outlined a proposal to build the communication system, bearing in mind the reuse and integration of IT GRC platform components, allowing the implementation of tools to support the whole IT GRC processes and taking profit from the integration of different components.
This proposal makes use of blackboard architecture to integrate different heterogeneous distributed components by means of tuple spaces and component-based environments to build a fully extensible user interface.

REFERENCES

[1] Rasmussen, M., Kark, K., Penn, J., McClean, C., Bernhardt, S.: Trends 2007: Governance, risk and compliance: Organizations are motivated to formalize a federated GRC process. Technical report, Forrester Research (April 2007)
[2] Nicolas Racz, Edgar Weippl, Andreas Seufert “A process model for integrated IT governance, risk, and compliance management” Databases and Information Systems. Proceedings of the Ninth International Baltic Conference, Baltic DB&IS 2010. Riga: University of Latvia Press, pp. 155-170.
[3] M.N. Kooper, R. Maes, E.E.O. RoosLindgreen “On the governance of information: Introducing a new

https://dx.doi.org/10.22161/ijaers/3.12.1 *ISSN: 2349-6495(P) | 2456-1908(O)*

concept of governance to support the management of information". International Journal of Information Management: The Journal for Information Professionals,Volume 31 Issue 3, June, 2011, Pages 195-200

[4] P. C. B. de Oliveira, N. F. da Silva, and M. M. da Silva, "A process for estimating the value of itil implementations," in Conference on Enterprise Information Systems (CENTERIS 2009). Springer-Verlag, October 2009.

[5] G. Hardy, "Using it governance and cobit to deliver value with it and respond to legal, regulatory and compliance challenges," Information Security

[6] Kuppuraju, S., A. Kumar, and G.P. Kumari, Case Study to Verify the Interoperability of a Service Oriented Architecture Stack, in International Conference on Services Computing. 2007, IEEE: India.

[7] Raja, M.A.N., H.F. Ahmad, and H. Suguri, SOA Compliant FIPA Agent Communication Language. 2008, IEEE.

[8] Battle, R., & Benson, E. Bridging The Semantic Web and Web 2.0 with Representational State Transfer (REST). Web Semantic, Vol. 6, 61-69, 2008.

[9] Kao, Y.-C. and M.-S. Chen, An Agent-Based Distributed Smart Machine Tool Service System, in 3CA 2010. 2010, IEEE.

[10] Parkin, S., D. Ingham, and G. Morgan, A Message Oriented Middleware Solution Enabling Non-repudiation Evidence Generation for Reliable Web Services, in ISAS 2007. 2007, Springer: Verlag Berlin Heidelberg p. 9– 19.

[11] CHUSHO, T. and K. FUJIWARA, A Formbased Agent Communication Language for Enduser-Initiative Agent-Based Application Development. 2000, IEEE.

[12] M. O. Shafiq, Y. Ding and D.Fensel, Bridging multi agent systems and web services: Towards inter-operability between software agents and semantic web services, Proc.of the 10th IEEE International Entreprise Distributed Object Computing Conference,PP. 85-96, 2006.

[13] E.Cerami, web services Essentials Distributed Applications with XML-RPC, SOAP, UDDI &WSDL, 1st Edition, O'Reilly & Associates, Inc., 2002.

[14] Y.LI, W. Shen and H. Ghenniwa, Agent-based web services framework and development environment, Computational Intelligence,vol.20, no.4, PP. 678-692, 2004.

[15] X.Liu, A multi agent based service oriented architecture for inter-entreprise cooperation system, Proc. Of the 2nd International Conference on Digital Telecommunication, pp.22,2007.

[16] S.Elhasnaoui, H. Medromi, A.Sayouti, Multi-agents modeling platform for IT governance based on ITIL" International Conference on Engineering Education and Research , ICEER 2013.

[17] S.Elhasnaoui, H. Medromi, S. FARIS, H.IGUER, A. Sayouti "Designing a Multi Agent System Architecture for IT Governance Platform" International Journal of Advanced Computer Science and Applications IJACSA Volume 5 Issue 5 May 2014.

[18] H.IGUER, H. Medromi, S.Elhasnaoui, S. FARIS, A. Sayouti «The Impact of Cyber Security Issues on Businesses and Governments- A framework for implementing a Cyber Security Plan» International Symposium on InterCloud and IoT -ICI Symposium 2014.

[19] T; Aihkisalo, T.Paaso "Latencies of Service Invocation and Processing of the REST and SOAP Web Service Interfaces" 2012 IEEE Eighth World Congress on Services

[20] A.Sayouti, H. Medromi, Book Chapter in the book "Multi-AgentSystems - Modeling, Control, Programming, Simulations andApplications", ISBN 978-953-307-174-9, InTech, April4, 2011.

[21] Shoham, Y. Agent-oriented programming. Artificiel Intelligence, February 1992. Stanford,USA.

[22] J. Ferber, "Les systèmes multi-agents, vers une intelligence collective", InterEditions, 1995, pp. 63-144

[23] Y. Sekhara,H.Medromi, "Multi-agent architecture for decision about the best IT management practices to implement" September 2015, UNET 15