Service Integration Engineering: A Paradigm Shift in Software Engineering

Farzad Sanati*
The American University of Iraq – Sulaimani, Kurdistan Region of Iraq.

* Corresponding author: Tel: +964-53-511-1959; email: Farzad.sanati@auis.edu.krd

Manuscript submitted October 6, 2015; accepted December 25, 2015.
doi: 10.17706/jsw.11.4.431-439

Abstract: The engineering field has taken on many new disciplines as our scientific knowledge has grown. The latest discipline in this is software engineering. Service Oriented Architecture, and in a broader sense, Service Oriented Computing have influenced Information and Communication Technology towards a design of uncoupled yet coherent architectures of services. Hence, it would be fair to say that Service Oriented Computing is software development with services where existing services are composed and configured to create new composite services and applications. The basis for service composition is often a workflow, which is a logical sequence of activities that, together, model a coherent business process. As a result of that, a new engineering paradigm has been emerging from within the software development trend towards the Service Oriented Computing, which in this paper referred to as Service Integration Engineering. This paper is intends to identify distinctive attributes of such paradigm and the circumstances that may have been instrumental in giving rise to this new paradigm. We intend to establish whether the past and current trend towards building software applications or larger systems entirely from pre-defined building blocks calls service does in fact represent a paradigm shift in software engineering or it is just a temporary methodological shift to use certain pattern that would soon vanish with the emergence of other patterns.

Key words: Software engineering, service oriented computing, service integration engineering, basic business service.

1. Introduction

Although it is commonly perceived that Web Services are technology solutions designed to add agility to business processes, experience shows that technology solutions rarely deliver agility unless they are primarily focused on business objectives. Technology vendors claim to add agility to the electronic service development process, trying to sell integration and development tools which mostly focus on 'Business Process Orchestration' or 'Web Services Workflow' [1]. They use a technology-driven approach that essentially misses the main point of the services. Arguably, as always with IT, the focus falls on technology rather than methodology.

Early literature [2] has investigated existing component-based agile software methodology to identify the gaps for web services development. A traditional component-based software development approach based on waterfall methodology (with some minor variations in different literature) consists of five distinguished phases, namely Requirements, Analysis, Design, Implementation and Testing. Although each phase is intended to be independent, there is always the possibility of overlap as well as iteration between the phases to ensure the correct design and complete implementation of the requirements. Artefacts or outputs are produced at the end of each phase, which in turn become input of another phase and also becomes the benchmark for a test of the activities of the same level.

This paper uses the combination of literature review and industry field survey to collect data and conduct statistical analysis in order to evaluate whether the past and current trend towards building software applications or larger systems entirely from pre-defined building blocks calls service in fact represent a paradigm shift in software engineering or it is just a temporary process shift to use certain pattern such as Service Oriented Architecture (SOA). If this is to be a paradigm shift, how could it affect the way we develop applications. The rest of this paper is organised as follows: Section 2 is the review literature regarding relevant topics on SE, is this section we also conduct quantitative analysis on the software industry practice in regards to the development and the use of web service and its practical impact on development methodologies. Section 3 presents the main arguments of this paper, first coining the term Service Integration Engineering (SIE) second to see how the move towards SIE is in fact a paradigm shift, hence to define the parameters and specifications of the new paradigm. Section 4 presents the conclusions of this paper

2. Literature and Survey Review

Software Development Life Cycle (SDLC) models are designed to guide the development activity to correctly follow a series of steps in creating software to meet business needs. The SDLC models have evolved as new technology and new research has addressed the weaknesses of older models. Ideas have been borrowed and adapted among the various models.

Integrating existing or newly developed atomic services could be a challenging engineering task because they are different from developing traditional software applications. This paper compares the generic classical Software Engineering (SE) tasks and activities including the proposed SOA extension [2, 3] with activities that we presume to be specific to SIE in order to represent the differences from a comparative perspective. In order to achieve this objective we first review SIE enabling models and technologies.

2.1. Web Services Technology

Service Oriented Architecture [4], and in a broader sense, Service Oriented Computing [5] (SOC) have influenced Information and Communications Technology (ICT) towards a design of uncoupled yet coherent architectures of services. Current ICT industry trends indicate that organisations that have large and critical legacy systems are moving towards the decomposition of legacy complex processes into atomic and simpler components to handle the ever-increasing complexity of current information [6]. This trend has led to a two-phase solution: Phase 1 is to transform massive architectures into constructs, consisting of simpler building blocks, called services; and Phase 2 is to recompose these services into complex services in order to achieve added value. This massive effort is simply the result of a new view in the application and systems development market that has been promoting the idea of using simpler and more generic building blocks to develop large and complex systems similar to those in civil engineering where they use simple breaks to build massive structures.

2.2. Software as Service

Web services interact with one another dynamically and use Internet standard technologies, making it possible to build bridges between systems that otherwise would require extensive development efforts. Traditional application design depends upon a tight interconnection of all subordinate elements, often running in the same process. The complexity of these connections requires that developers comprehensively understand and have control over both ends of the connection; moreover, once established, it is extremely difficult to extract one element and replace it with another. By contrast with tight coupling principles that require agreement and shared context between communicating systems as well as sensitivity to change, loose coupling requires a much simpler level of coordination and allows for more flexible reconfiguration [7].

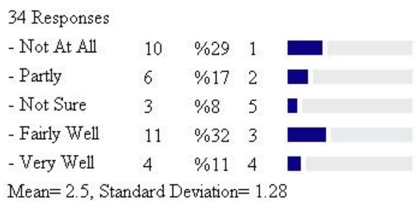
For SOA to work, it is not enough to build and deploy a collection of systems and services. Services are meant to be shared, which means they must be created according to certain rules that everyone can follow. The collation of related rules is known as 'standards', without which an SOA cannot function. Web services are also defined by an interface that can be formally stated using the Web Service Description Language (WSDL) [8], which defines and advertises the functions and behaviours provided by the Web service [9].

2.3. IT Industry Survey

An important area that this research needed to clarify was the level of involvement (knowledge and experience) of the Information Technology (IT) industry in service integration. An online industry survey by "DeSI" laboratory [10] at the University of Technology, Sydney (UTS) has gathered and analysed information from practicing software engineers, project managers and developers to determine the common practices and tools used in performance of their jobs. This survey was particularly designed to discover the methodologies used in web services development and composition. It gives the data in order to determine not only how these methods differ from Object Oriented Development (OOD) methods but also the patterns used in SOA development indicating a drive in paradigm shift from SE to SIE. The results of the survey are not conclusive; nevertheless the information obtained survey could be used to direct further research into development and fine-tune the conclusions of this research.

A total of 40 survey participants were selected from entirely different industries, areas of work and responsibilities within the ICT industry.

According to the statistics obtained in this survey, it appears that most of the ICT industry in the surveyed sector is one way or another practicing service development. In the question of being familiar with SOA, more than 60% of participants have indicated that they are in use of SOA. (See Fig. 1).

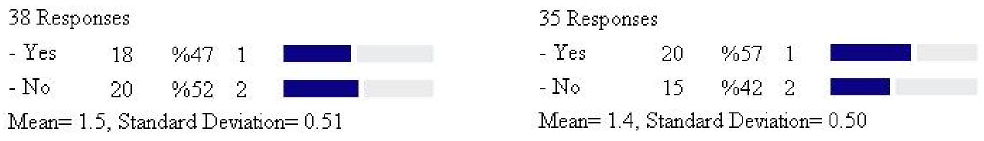

34 Responses

- Not At All	10	%29	1
- Partly	6	%17	2
- Not Sure	3	%8	5
- Fairly Well	11	%32	3
- Very Well	4	%11	4

Mean= 2.5, Standard Deviation= 1.28

Fig. 1. Familiarity to SOA.

On the question of using SOA, 47% of organisations have indicated the use of SOA in software development of some kind. However one out of 10 developers says they have no or very little documentation on their development practice. On the other hand 57% of the IT professionals confirmed that their organisation is developing Web services. (See Fig. 2)


38 Responses

- Yes	18	%47	1
- No	20	%52	2

Mean= 1.5, Standard Deviation= 0.51

35 Responses

- Yes	20	%57	1
- No	15	%42	2

Mean= 1.4, Standard Deviation= 0.50

Fig. 2. Using SOA and developing web services.

Although 57% of IT managers confirmed their organisation is developing web services, only 4% are familiar with any formal Service Oriented Architecture or design patterns. Only 27% of the survey participants who are developing web services are conducting some form of interoperability analysis for service composition.

Analysis of the aforementioned industry survey data, specifically the last two points, in conjunction with the cited literature are evidence of the industry adoption of recent results in the area of web services development. The case may differ from industry to industry, however we stress the need for more research on modelling frameworks for service integration and interoperability analysis. Section 3 is an attempt to present a comparative analysis of Software Engineering with Service Integration Engineering from both technical and methodological (process) view points by facts and data obtained from the literature review and the industry survey.

2.3. SE vs. SIE Comparative Analysis

Some interoperability analysis and provisions have become necessary SE tasks in recent years, mostly due to the distributed nature of modern business operations. This fact highlights the importance of essential interoperability requirements for distributed databases and applications, however, this interoperability analysis and design has never been an integral part of the software development lifecycle and has only been performed in an informal manner as technical analysis activities fitted in between other tasks in all phases of the SDLC. As illustrated in Table 1, every phase in a classical SE lifecycle has a corresponding phase in the presumed SIE lifecycle although they perform different tasks and activities to produce appropriate outcomes for each phase.

2.4. Basic Business Service

Basic Business Service (BBS) is a concept that we use as guiding metaphor for customer-centric service provision. From service integration point of view, a “BBS is a collection of actions including at least one service, which when executed in its appropriate workflow fulfils a business function”. This section explains the principles used for the analysis and modeling of BBS.

The concept of abstraction in object-oriented paradigm[11] plays an important role in the representation of complex data structures. Abstract objects or data structures can form hierarchical representations to provide easy-to-understand solutions for complex models. Abstraction is the means by which only a certain level of information detail is exposed by the entity, depending on the levels of representation intended for that model.

Different levels of data abstractions are also known as the level of granularity in a model. This study invokes the principle of data abstraction in context of Basic Business Service (BBS) to represent an integrated service in different levels of granularity, depending on the detailed information about its underlying service structure and business rules. The BBS model effectively supports the concept of electronic service integration by combining basic services offered by multiple service providers into a single complex service that corresponds to a unique business function. We can also use the concept of BBS to be the building blocks of yet more complex services to act as an integrated complex business function called Composite BBS. These concepts are illustrated in Fig. 3.

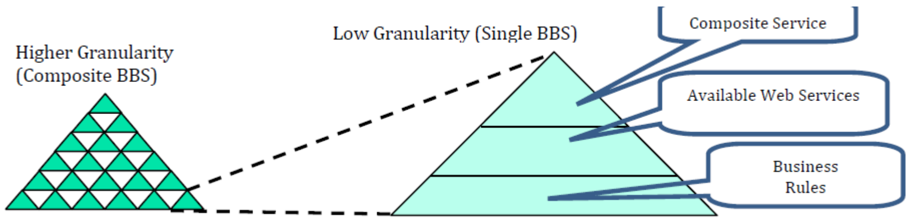

Higher Granularity (Composite BBS)
Low Granularity (Single BBS)
Composite Service
Available Web Services
Business Rules

Fig. 3. BBS and composite BBS as the building blocks of integrated complex business service.

BBS has the capacity to organise the way atomic services are analysed, modelled, composed and delivered to provide a customer-centric complex service. An important area that this research needed to understand was the level of involvement (knowledge and experience) of the Information Technology (IT) industry in service integration [12]. An online industry survey by this study considers the concept of BBS to be almost equal to the concept of 'composite service' from the technical prospective, and we use the words *BBS* and *composite service* interchangeably. (Examples of BBS are: *applying for a loan*, *buying an airplane ticket* or *buying a house*.)

2.5. Service Engineering Processes

Table 1. Comparing Processes between SIE and SE

SE Phases	**Software Engineering (SE) tasks**	**Service Integration Engineering (SIE) tasks**	**SIE Phase**		
Analysis	Identify business requirements and translate them into functional and non-functional requirements.	• Identify the main service requirements to fulfill the business function. • Identify Quality of Service (QoS)	Service Identification	**BBS Identification**	**Business Requirements Identification**
	N/A	Identify atomic service interoperability requirement at semantic and syntactic levels.	Interoperability Identification		
Design	Translate the requirements into conceptual models. Analyse requirements to define high-level design structure.	Select existing and/or design new atomic services that can play a role in the main business function.	Single BBS Design	**BBS Design**	
	N/A	Design the atomic service interface contracts.			
	Design major system components (use design patterns).	Construct a design model for the service group (BBS) from available atomic services.			
	Describe the responsibilities of all system objects and their relationships to other components.	Design and configure the relationships between all BBSs with service providers and service consumers.	Complex BBS Design		
	N/A	Design composite BBS (the composition of BBSs) Architecture			
Delivery	Business component implementation	Service Implementation	BBS Implementation	**BBS Delivery**	
	System data implementation.	Workflow Implementation			
		Interoperability configuration	BBS Test and Configuration		
	System / user acceptance Test	Workflow / user acceptance test			

Three main areas of focus in the literature were recognisable while reviewing research works related to the concept of service integration engineering as follows: 1) Service development/creation [13], [14], where they concentrated on how to create web service from the ground up in order to preserve interoperability and availability of business functions over the internet [2], [15]. 2) Service integration/composition [16], where authors provide road maps for breaking down large legacy business functions in to smaller pieces then creating services out of those in order to be integrated. Moving forward in this area, we see more shift towards understanding the importance of how we can use proven successful patterns to achieve a systematic modelling of entire systems built on services [17], and 3) Service engineering frameworks and processes [16], [18], [19], where they are advocating methodological (process oriented) approach to the whole processes of building new system entirely out of services (SOA).

Table 1 summaries our compertive analysis between the SIE and the classical software engineering process described in many literature starting with Sommerville [20] and continued with many others literature [21], [22]. This comparison, which is based on the aforementioned literature reviles very little similarities between the two paradigm (SIE and SE), where the greatest emphasis in SIE is on solving complex interoperability problems and service workflow configuration rather than designing new components and finding the best ways of wire them together using patterns such as MVC [23]. We see substantial differences between activities performed in the SIE project and a typical SE project that uses component-based software development methodology for example. (See Table 1).

Service integration engineering if any should be about configuring the interoperability of existing services to create new composite web services and workflows so they can deliver to customers better and more complex services. Table 1 illustrates that most tasks in SIE are interoperability-related.

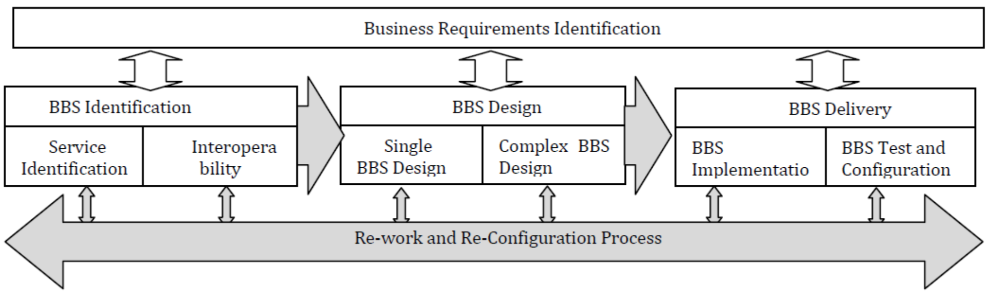


Fig. 4. BBS concept within the SIE process.

Interoperability specification and enabling tasks seem to have not been previously considered as an integration-specific stage in any SDLC, yet industry agrees [10] that these two steps are crucial to any successful service integration project. Important new concepts such as basic business service in the context of service delivery [24], which are described as client's basic service requests, such as applying for a loan, are also new to the classical software engineering. Diagram in Figure 4 is the graphical illustration of the roll of BBS in SIE process.

- Newly-introduced concepts need to be more thoroughly analysed in a context of service integration in general and in web service integration specifically. A comparative analysis of both processes detailed in Table 1 can be summarised as follows:
- Basic Business Service (BBS) is considered to be the fundamental unit of requirement for service integration projects, as opposed to 'basic function' which is the simplest unit of requirements in traditional software development process. In a broader sense, from the service consumer's point of view, it comes down to using one or more services to satisfy a typical service consumer's need.
- The simple fact that the complexity of service integration is rapidly growing as a result of the constant increase in available services further highlights the need for a formal model approach and design standards to ensure efficient - and more importantly, repeatable - service integration process. The substantial difference between the two processes provides enough reason to believe that the traditional software development frameworks are not only ill-equipped for efficiently dealing with service integration projects, but also technical complexity issues may increase the risk of complete or partial project failures.

Repeatability is the most important and key attribute of modern engineering frameworks. From a service integration point of view, it is essential to identify services and their interoperability constrains in order to drive a trust worthy integration framework standard.

3. Service Integration Engineering Explained

This section discusses the details of the main activities proposed in SIE process. It is important to note that services integration process is meant to be live and subject to change through feedback within the process. This feedback mechanism is facilitated through a “Re-work and Re-configuration process” that spans alongside of the main SIE process. Based on the discussions in Section 3 it becomes apparent that additional activities and techniques are required to handle the complexity and unique attributes of developing BBS and complex BBS. The rest of this section discusses the specific phases of SIE with the reference to the traditional SE process.

3.1. BBS Identification

Designing applications purely out of services requires the use of specific design and modelling techniques. There are two clear tasks in the BBS Identification phase, each task responsible to identify an important aspect of an atomic service: (1)The first task is to identify atomic services for BBS participation. During the process of this task business requirements of the application are evaluated to discover the atomic service requirements as potential building blocks for a BBS. (2) The second task is to identify and indicate the boundaries (scope). The purpose of this task is to identify atomic service interoperability requirement at semantic and syntactic levels.

The results of this phase are two artefacts: (1) A catalogue providing precise information on all the technical communication and QoS parameters of services provided by their interface or contract in some case WSDL. The role of this catalogue is essential for the BBS in the delivery phase, because it provides descriptors for every atomic service participating in the BBS. (2) Specification of all the services in the form of BBS Services. This specification is in the form of data file designed to provide all the knowledge required to understand the higher level of requirements for the proposed complex BBS.

3.2. BBS Design

Workflow modelling and design can further be divided into generation and specification stages [25]. Workflow models are considered to be *manual, semi-automatic* or *automatic* depending on the level of semantic and dynamic knowledge representation of the model. The specification of the model, depending on the level of semantic intelligence representation, can be implemented using industry standards such as Business Process Engineering for Web Services (BPEL4WS) or OWL-S. Industry standards such as BPEL4WS are more suitable for static workflows with no semantic intelligence and are entirely configured at design time; however standards such as OWL [26] handle specific semantic information that could be used to design a dynamically configurable workflow *(automatic and semi-automatic models).*

A unified and repeatable service integration process must make use of best practice modelling techniques in a way that increases its reusability in different scenarios, even if these scenarios are in different implementation domains. Our goal in modelling and design is to visualise the BBS design in various levels of abstraction at various phases of SIE process. Hence the following tasks are identified for BBS Design phase:

1) The first task in BBS design phase is to prepare knowledgebase artefacts that can provide knowledge about a lower granular level of an abstract BBS. This task must produce a data file that carries all the knowledge required by the system to understand the various details of a BBS model in order to deliver a specific instance of that BBS workflow. Therefore make it possible to create a single BBS.
2) The second task in design phase is the complex BBS workflow design, which ensures that every business rules requirement of every atomic service participating in BBS is understood and enforced as a rule by the designer. If we have chosen a number of web services to participate in a BBS, we must ensure that they are arranged in such a way that there are no prerequisite loopholes or conflicts

between services. For example, if there are three possible web services involved in our *Loan Application* BBS, we must ensure that they are arranged in such a way that the prerequisites of all the services are satisfied when the BBS is executed.

One of the most important areas of analysis and design discussed in this study relates to business rules and regulations. These rules and regulations are to be organised in form of data files to illustrate the semantic correlation of every element in the BBS. One way to organise the business rules knowledge is to imbed them in an OWL-S ontology data file and attach it to the BBS model when it is prepared and delivered as an item of results from the tasks of this phase.

3.3. Delivery Phase

Delivery phase is simply the implementation of all the design decisions. Process of identifying the business requirements of the application is illustrated in Figure 4 as a parallel task alongside of the main SIE process; this is to ensure the continuous validation of artefacts in every stage against the business requirements of the application. The Delivery phase mainly deals with constructing alternate workflow instances that are produced based on semantic information extracted from OWL-S and information provided by the service consumer. Our strategy is to facilitate the seamless evaluation of composition candidate services; to improve composability for run-time workflow reconfiguration.

4. Conclusion

The results and findings of this paper are by no means conclusive, but what is definitive about the findings of this research is the need for paradigm shift in software engineering process that can reflect and accommodate the practical paradigm shift in software development industry, and this paper is a logical response to this shift. In this study we put forward our findings to indicate that creation of composite service architecture must focus and represent business objectives, which often concern producing an added value by delivering a better and more reliable service for users of those composite services.

The complexity of integrated web services, especially those developed up to the advanced transactional stage requires a highly disciplined and repeatable process approach to ensure the most efficient and reliable organisation of services towards an integrated BBS driven system. In this paper, Service Integration Engineering process was proposed and discusses, which can accommodate the integration-specific tasks into a typical design, build and delivery process from a purely SOA prospective.

References

[1] Arroyo, S., Sicilia, M.-A., & Dodero, J.-M., (2006). Choreography frameworks for business integration: Addressing heterogeneous semantics. *Computers in Industry*.

[2] Siew, P. L. L. & Eng, P. C. W. L. (2006). Web services implementation methodology for SOA application. *Proceedings of the IEEE International Conference on Industrial Informatics*.

[3] Oracle. Software engineering in an SOA environment. Retrieved May 2, 2011, from http://www.oracle.com/technetwork/topics/entarch/oracle-pg-soa-sw-engineering-r3-0-176714.pdf

[4] Zimmermann, O., Krogdahl, P., & C. Gee. Elements of service-oriented analysis and design. Retrieved May 30, 2008, from http://www-128.ibm.com/developerworks/webservices/library/ws-soad1/

[5] Turner, M., Budgen, D., & Brereton, P. (2003). Turning software into a service. *Computer*, *36(10)*, 38-44.

[6] Huhns, M. N., & Singh, M. P., Service-oriented computing: Key concepts and principles. *IEEE Internet Computing*, *1(1)*.

[7] Papazoglou, M., Web services technologies and standards. *Computing Surveys*.

[8] Chinnici, R. *et al*. Web services description language (WSDL) version 2.0 Part 1: Core language. 2 Retrieved September 2, 2011, from http://www.w3.org/TR/wsdl20/

[9] Gudgin, M., Hadley, M., & Rogers, T. (2006). W3C web services activity. Retrieved November 16, 2010, from http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/

[10] Sanati, F. (2010). Software industry practice survey. Retrieved November 5, 2010, from http://farzadsanati.blogspot.com/p/software-industry-survey.html

[11] Kramer, J., & Hazzan, O., The role of abstraction in software engineering. *Proceedings of the 28th International Conference on Software Engineering*.

[12] Newton, *Gravity*, London: Oxford University.

[13] Turner, M., Budgen, D., & Brereton, P. (2003). Turning software into a service. *Computer*, *36(10)*, 38-44.

[14] Fensel, D., & Bussler, C. (2002). The web service modeling framework WSMF. *Electronic Commerce Research and Applications*, *1(2)*, 13-137.

[15] Zhao, J. L., Tanniru, M., & Zhang, L.-J. (2007). Services computing as the foundation of enterprise agility: Overview of recent advances and introduction to the special issue. *Information Systems Frontiers*.

[16] Shen, W., *et al*. (2010). Systems integration and collaboration in architecture, engineering, construction, and facilities management: A review. *Advanced Engineering Informatics*, *24(2)*, 196-207.

[17] Paraskevopoulou, M. D., *et al*. (2013). DIANA-microT web server v5. 0: service integration into miRNA functional analysis workflows. *Nucleic acids Research*.

[18] Biffl, S., & Schatten A. (2009). *A Platform for Service-Oriented Integration of Software Engineering Environments*.

[19] Arni-Bloch, N., & Ralyté, J. (2008). Service-oriented information systems engineering: A situation-driven approach for service integration. *Advanced Information Systems Engineering*.

[20] Sommerville, I. (2004). Software engineering.

[21] Jayasinghe, A. J., & Weigand, H. (2012). Business service integration using pattern composition. *Advanced Information Systems Engineering Workshops*.

[22] McConnell, S. (2006). *Software Construction, 23(2)*, c3-c3.

[23] Sunmicrosystems. Retrieved April 23, 2011, from http://java.sun.com/blueprints/patterns/MVC-detailed.html

[24] Castellano, M. (2005). An e-government cooperative framework for government agencies, *Proceedings of the 38th Hawaii International Conference on System Sciences*.

[25] Liu, W., Husni, H., & Padgham, L. (2007). E-service composition tools from a lifecycle perspective. *E-Service Intelligent*.

[26] McGuinness, D.L. Retrieved 2004, from http://www.w3.org/TR/2004/REC-owl-features-20040210/

Farzad Sanati is a Kurdish Australian Academic, who received his PhD in computing science from the University of Technology Sydney in 2011. He received his masters of software engineering from the University of Western Sydney Australia in 2004. Dr Sanati is currently the chair of Institutional Review Board (IRB) as well as his teaching role as an assistant professor of Information Technology at the American University of Iraq, Sulaimani. He was a program committee member of the sixth International Conference on Digital Society (ICDS 2012), Valencia Spain and the International Conference on E-commerce and Web Technologies (EC-WEB 08), Milan Italy. In addition to his academic and research work, he has extensive experience in many major E-Government and ICT projects in Australia. In addition, He has been giving advice to Chinese Government officials from e-Government and ICT Department of Jiangsu Province (the hub of information technology industry in China).