1-1-2014

Aligning ontology-based development with service oriented systems

Jun Shen
University of Wollongong, jshen@uow.edu.au

Ghassan Beydoun
University of Wollongong, beydoun@uow.edu.au

Graham Low
University of New South Wales, g.low@unsw.ed.au

Lijuan Wang
University of Wollongong, lw840@uowmail.edu.au

Aligning ontology-based development with service oriented systems

Abstract

This paper argues for placing ontologies at the centre of the software development life cycle for distributed component-based systems and, in particular, for service-oriented systems. It presents an ontology-based development process which relies on three levels of abstraction using ontologies: architecture layer, application layer and domain layer. The paper discusses the key roles of ontologies with respect to the various abstraction layers and their corresponding impact on the concomitant workproducts. In addition, a peer-to-peer-based service selecting and composing tool is suggested as a way of supporting the process. The paper presents the architecture of the proposed tool and illustrates the whole process in the development of a mobile banking application based on dynamic Web services.

Keywords

ontology, service, aligning, development, oriented, systems

Disciplines

Engineering | Science and Technology Studies

Publication Details

Shen, J., Beydoun, G., Low, G. & Wang, L. (2014). Aligning ontology-based development with service oriented systems. Future Generation Computer Systems: the international journal of grid computing: theory, methods and applications, 32 263-273.

Aligning ontology-based development with service oriented systems

Jun Shen[1a], Ghassan Beydoun[1], Graham Low[2], Lijuan Wang[1]

[1]*School of Information Systems and Technology, University of Wollongong, Wollongong, NSW, Australia*
{jshen, beydoun}@uow.edu.au, lw840@uowmail.edu.au
[2]*Australian School of Business, The University of New South Wales, Sydney, NSW, Australia*
g.low@unsw.edu.au
[a] *Corresponding author: Tel: +61-2-42213873, Fax: +61-2-42214045*

Abstract

This paper argues for placing ontologies at the centre of the software development lifecycle for distributed component-based systems and, in particular, for service-oriented systems. It presents an ontology-based development process which relies on three levels of abstraction using ontologies: architecture layer, application layer and domain layer. The paper discusses the key roles of ontologies with respect to the various abstraction layers and their corresponding impact on the concomitant workproducts. In addition, a peer-to-peer-based service selecting and composing tool is suggested as a way of supporting the process. The paper presents the architecture of the proposed tool and illustrates the whole process in the development of a mobile banking application based on dynamic Web services.

Key words: *Software Development Lifecycle (SDLC), Ontologies, Agents, Multi Agent Systems (MAS), Peer-to-Peer (P2P) Systems, Service-Oriented Systems*

1. Introduction

A service oriented architecture (SOA) promotes loose coupling between components to enable faster and more flexible reconfiguration of business processes and provides a means of organising system resources in an open and flexible arrangement. From an enterprise management perspective, the successful delivery of service oriented architecture (SOA) systems translates into responsive business processes that can adjust to varying customer service requirements [9, 11]. Expected responsiveness and adjustment is based on leveraging the knowledge of relationships between various services and mixing and matching groups of services to satisfy new requirements. Service oriented computing (SOC) has become a convincing paradigm for enterprises in tackling traditional hurdles to improving the efficiency and effectiveness of their ubiquitous software applications [13]. Vice President of Gartner, J. Fenn, predicted that by 2013, SOA will have delivered transformational results to the role and capabilities of IT for businesses [14]. Furthermore in 2012, one third of IT budgets were spent gaining access to services developed by other vendors [15]. This increased attraction to SOC comes with the expectation that energies from software development and acquisition will be shifted to other business activities, and at the same time deliver better alignment with business requirements.

Service-oriented software engineering promises to deliver enormous tangible improvements to business process enablement. In practice it has been hard to realize, especially for complex application systems, when it is advantageous to use services from various providers. There is a need to find ways to collect available business services from the different providers and to specify how such a collection of services should be combined and integrated seamlessly [26]. This ability to easily integrate services can increase flexibility and agility, not only in systems development but also in business process management. However, existing service components do not provide a clear and comprehensive definition of the business process semantics. Therefore many existing services are often isolated and opaque to information system developers. Current software techniques and tools do not alleviate this and place too much burden on developers attempting to reuse existing services. This hampers the realisation of the monetary benefits of the technology and the collective adoption of reuse by the required large number of players to ensure a critical mass of shared services [23].

The use of ontologies can pave the way for an intelligent software development environment where developers submit new business requirements and an automatic tool generates the service oriented software system. With semantic driven composition, services will be shared between teams of developers and across multiple organisations connected via the Internet. We are acutely aware that existing Web languages are not easily accessible as described in Berners-Lee [2]. To overcome this, we aim to provide ontological support to the requirements analysis phase to ensure that any newly created service can be appropriately indexed by a semantically rich layer of ontologies. There are a number of promises and arguments around semantic Web services [28, 29, and 36], one big issue being the limited generality of the isolated efforts made by independent

research groups, for example, as reported in [32]. Indeed, this paper heeds visionary comments in [24] and [17] that, in order to meet the requirements of pervasiveness and autonomy in services development, the maturity and awareness of semantic Web and ontology technologies (key components of Web 3.0) should not be overlooked. This research uses an innovative ontology- and agent-based technology to support the SOA environment by providing automatic identification and merging of services which will in turn enable process changes in the business requirements.

The holistic semantic Web driven SOA development approach will play a significant role in better exploitation of services at both the technical and business level. This paper is well timed coinciding with the rapid development of Web 2.0 technologies which enable a single enterprise to use the Web to offer value-added customer services via connection to various services or applications from other public or commercial organizations. The innovations in this paper will harness the Web as a support medium for systems development enabling automatic exchange of services between various development teams across multiple organisations. The significance of these innovations is more pronounced coinciding with challenges of dynamic composition of distributed services [34] created by the high demand for portable appliances such as Personal Digital Assistants (PDAs) and next-generation mobile devices as well as for P2P applications e.g. [35,43]. It's challenging to apply a hierarchy of ontologies in developing such applications. For example, concerns remains about how to measure the quality of the ontologies and the alignments among them [5].

This paper proposes an intelligent and supportive software development environment where developers can focus on semantic enrichment of business requirements and proper alignment with business processes rather than the time consuming task of service identification and integration. This paper's contributions are: a holistic ontology based development approach for service oriented systems such Gird and Cloud platforms; a three layer abstraction of ontology alignment; and the introduction of a semantic integration life cycle for semantic Web services. Our proposed approaches were demonstrated by applying them in the building of QoS ontology for a service oriented system and developing a multi-agent, peer-to-peer based service system to align ontologies before tool and illustrating the process in the development of a mobile banking application based on dynamic Web services.

The rest of the paper is organised as follows: Section 2 provides an argument for placing ontologies at the heart of SDLC of component-based systems generally and service oriented development in particular. Section 3 articulates the requirements for an ontology-based service oriented development and the existing supporting ontologies from the literature. Section 4 discusses the architecture of the intelligent development environment which uses a MAS for peer-based Web service composition system. A case study highlighting the ontology-based development of bank loan approval service oriented system is presented in Section 5. Section 6 concludes with a summary and discussion of future work.

2. Ontology-Centric Service Orientation

It is often a complex task for developers to locate the appropriate service components to customise and integrate into their system. To reduce this cost and to automate much of the service selection and composition effort, we advocate an ontology-based approach that uses a semantically enriched representation of services and business requirements in order to enhance interoperability of services. A domain ontology can facilitate reuse of services undertaken across different areas (or industries). For example, the services for certain accounting practices may vary but only slightly across application areas. Such practices, if well documented using a domain ontology, can provide reusable services that can be adapted using appropriate application information. It is fair to say that the development of any IS system can benefit from a domain ontology and/or an application ontology with this being most evident to developers during the analysis phase. Such ontologies may be available from existing repositories (e.g. [8]) or a domain analysis yielding an ontology may be considered the first stage of developing the system (e.g. as proposed in [7] or in [41]). Some industries such as banking and finance are inclined to provide their own ontologies to enable speedier IS development.

Unfortunately, only a small number of existing methodologies include ontologies in their workproducts and processes. This support is generally confined to the early phases of the development (the *analysis* phase). For example, Girardi and Serra specify how a domain model that includes goal and role analyses is developed from an initial domain ontology in their methodology [14]. Another example [10] uses ontologies to mediate the transition between goal and task analyses. A better inclusion of ontologies into a development methodology permits the long term reuse of software engineering knowledge and effort and can produce reusable components and designs [40]. But first some of the challenges discussed in [3] and [4] need to be addressed. Various software components have different knowledge requirements and they relate to the application domain from various layers of abstraction. Components may be complementary and they may

have varying degrees of prescription to the domain requiring various degrees of adjustment to suit the domain. For example, a user interface will operate at a different level of abstraction to the application domain than say a component interfacing to a data mining agent or a database service. In other words, the degree of linkage between software components and analysis models depends on the nature of components themselves and how the components relate to the system as a whole. An ontology based approach for development therefore further complicates the analysis activities during development. Even if we assume a high degree of independence between various software components and the application domain, identifying and appropriately using ontologies in the development of components remains difficult. Elsewhere, e.g. in [41] the use of two ontologies, a domain ontology and an application ontology are advocated to guide the verification of requirement models. In this paper, we advocate for an additional ontology that describes the relations between the software components and the way they are structured in the system. For the development of a service oriented system, the use of this *architecture ontology* will in turn enable proper assumptions to be made with relation to the domain and application ontologies and how they can be related to service components. In doing this, we not only identify various roles of an ontology, but we also identify suitable layers of abstractions to describe a collection of relevant ontologies that can enable reuse of components and domain analyses (as proposed in Figure 1).

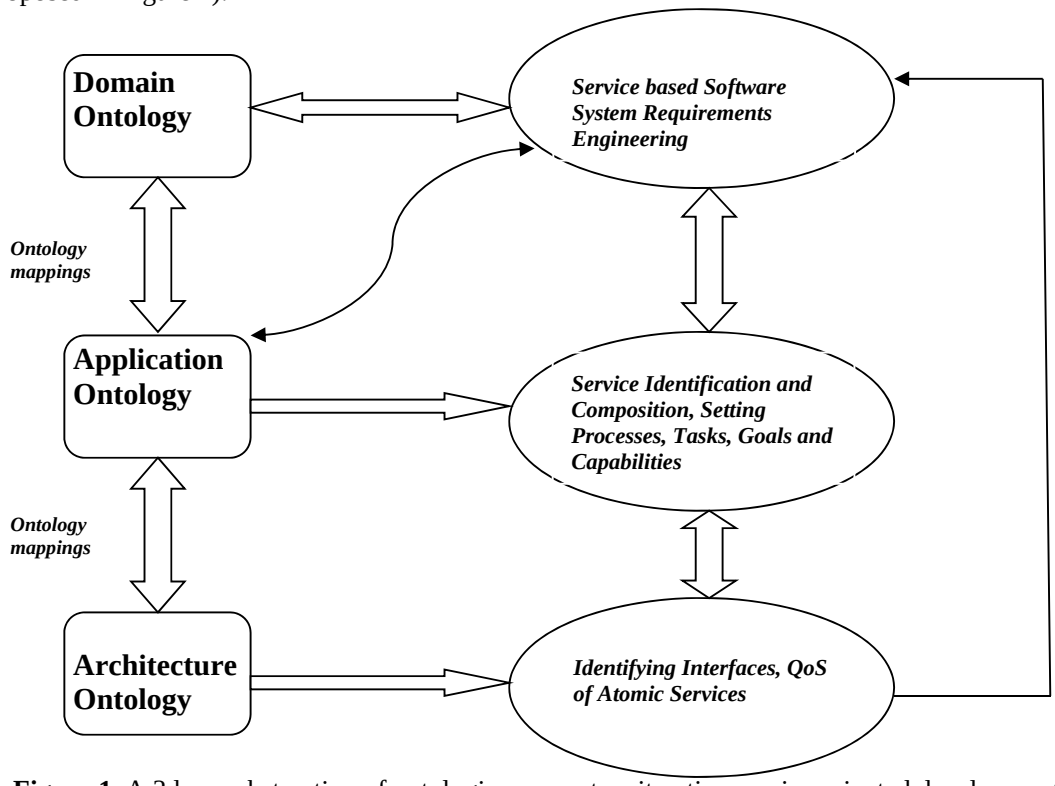

Figure 1. A 3 layer abstraction of ontologies support an iterative service oriented development lifecycle.

The Software Development Life Cycle (SDLC) requires the three related ontologies shown in Figure 1. The domain ontology describes the domain knowledge for services to be used in the development of the requirements for a solution to the problem. Domain ontologies may be unique to the problem itself or may be adapted from previous problems in similar domains. The application ontology describes the types of and nature of problems to which services have been developed. This ontology is usable to define the service capabilities and preconditions as well as indexing suitable services from any available service repositories. When various tasks in a software or business process can be implemented with a combination of services (old or new), service composition applies. In other words, the system may achieve the goal by incorporating previous knowledge of a similar previous problem. This composition can be automated when application requirements and service interfaces are aligned properly with the application. This is a key aim of our work. Finally, the architecture ontology describes the inter-services interfaces and how they connect. One important mechanism to identify suitable services at this level is to match the service requestor's quality of service (QoS) requirements and the service provider's QoS provision. Therefore, alignment of an QoS ontology for services at an architectural level is essential. Figure 1 also provides the methodological sketch accommodating the observations of this section.

We envisage that a service-oriented systems development starts with a domain ontology and an application ontology identifying tasks and functions required from the services in the system. These can index an appropriate set of services/capabilities from an appropriate existing library of services/capabilities. Chosen services do not necessarily have the required degree of application or domain dependence. The application ontology will assist in identifying the requirements of each service. Individual application ontologies for each service could be identified and used to verify chosen services against requirements. The architecture ontology will be used to create a common interface between the various services and glue them into a working system. This third ontology is essentially a service ontology that describes the knowledge required for control structure and service dependencies. An automatic tool that seeks to dynamically select a service (or its coded implementation) to solve a problem needs to know this ontology. The collection of all ontologies will form the basis for an iterative process to develop the architectural interfaces between all agents and verify the application requirements of chosen capabilities. The application ontology may also require incremental refinement during the iterative process. Appropriate ontology mappings are needed between application and architectural ontologies to facilitate service composition. Verification between services task capabilities and the architecture ontology is undertaken, which may result in further localized ontology mappings. The development process requires guidelines to develop the various ontology mappings between the three ontologies ensuring their completeness and consistency.

The ontology-based development will enable an ontology-supported approach to support dynamic composition of web services in a highly distributed and heterogeneous computing environment. The dynamic composition is adapted from [35] and highlights how ontologies can be exploited using semantically driven composition of services e.g. [38]). It is underpinned by reusing existing services and noting that they typically describe atomic tasks within a business process. The cloud and SOC pioneers such as Amazon and Google have developed vendor-specific Web APIs or categorical keyword-based platforms like EC2 (www.amazon.com/ec2)) and Google Play (play.google.com). They demonstrate convincing performance metrics such as scalability and elasticity, but on the other hand, they suffer from the lack of interoperability between different vendors or even different platforms (such as iOS and Android) for the same vendor. Typically they were only concerned about the application ontology as specified in our model without considering support for a domain ontology and an architecture ontology, so there are few mechanisms to describe non-functional requirements such as QoS or semantic relationships between various service provisions. Considerable research is occurring into the use of light weight ontologies, or namely linked data, to improve the scalability of semantic Web services [23, 32]. There are also approaches to bridging ontologies and traditional data models [1].

However, so far there has been little research into developing standardized ontologies, particularly for QoS of services for both the service requestor and the provider between services in a service oriented application. Notably, OWL-S (http://www.daml.org/services/owl-s/) is one effort to standardise the description of Web services with rich semantics. OWL-S and similar efforts are suitable to for reuse as architecture ontologies in our approach. However, as we will see later in this paper (section 4), they are not ready for direct use and they need to be properly aligned to the other ontology layers.

3 Supporting Ontology-based Service Oriented Development with Existing Architectural Ontologies

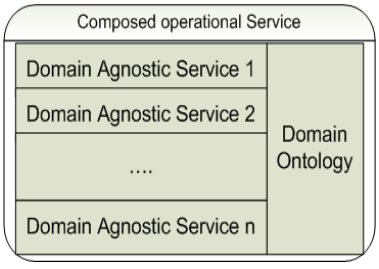

Figure 2. Ontologies can be used to give a dynamic interface to services

Our ontology-based approach complements existing service repositories which provide service implementations that may be used in both the design and implementation phases (Figure 2). Our use of ontologies at three levels, as discussed in Section 2, offers a unique workbench to test the reuse of ontologies placing them at the centre of the development lifecycle. This approach is motivated by existing suitable

ontologies. For instance, OWL-S can allow individual software agents to dynamically discover, invoke, compose and monitor Web services with a high degree of automation. It has also been delineated to easily identify tasks for individual services, ameliorating the difficulty of providing different knowledge requirements to different components. Another ontological engineering effort that can provide a starting point for an architecture ontology is WSMO. Aligning both of these as architectural ontologies is first described in this section, before using the QoS as a critical component at this layer to demonstrate the ontology-alignment process and enable a P2P automatic tool to support the ontology-based development.

3.1 Aligning OWL-S and WSMO for ontology-based approach

OWL-S ontology consists of three main components: the services profile, the process model and the grounding. The services profile is for advertising and discovering Web services. The process model is used to describe detailed operations of services and define composite Web services. The grounding is used to map the abstract definition of services to concrete specifications of how to access the services. The services profile component of the ontology can be detailed and refined to allow detailed services' description and evaluation. Basically, the service profile does not mandate any representation of services; rather, using the OWL subclass it is possible to create specialised representations of services that can be used as service profiles. OWL-S provides one possible representation through the class "Profile". An OWL-S "Profile" describes services individually as a combination of three basic types of information: what organisation provides each service, what functions each service computes, and the features that specify the characteristics of each service. In this way, the complementary descriptions about Web services including the QoS can be extended in the services profile, so that we can improve the automation and reliability of Web services' composition in a dynamic environment.

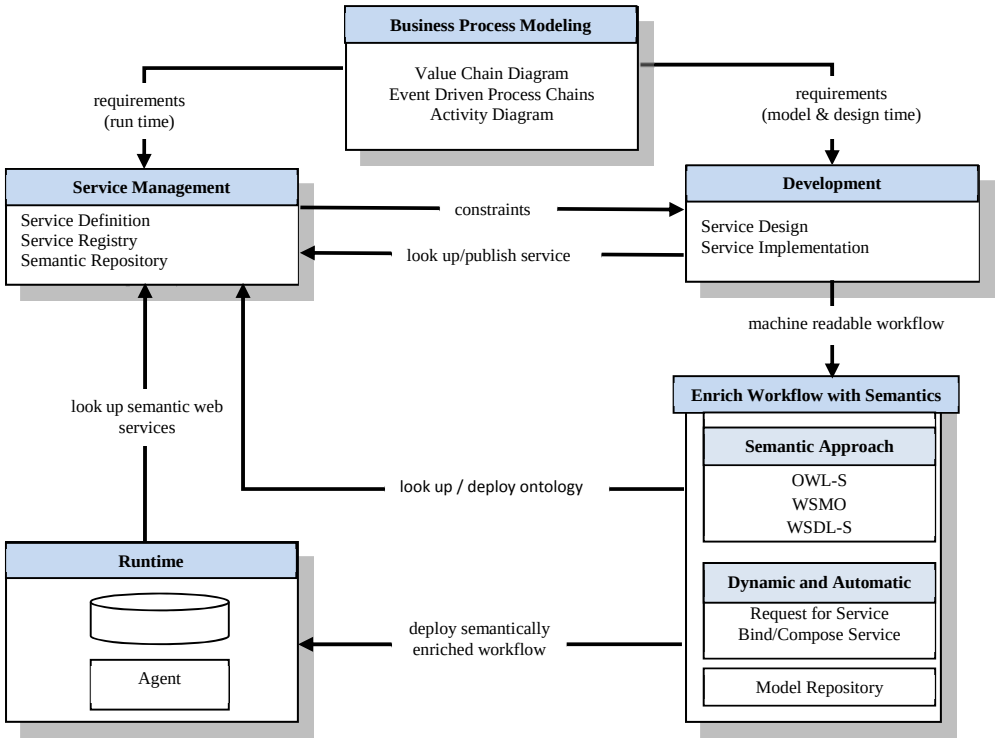

Figure 3. Semantic Integration Life Cycle for services development

WSMO defines four high-level notions that support the development of semantic Web services: namely Ontologies, Goals, Mediators and Web services [33]. *Ontologies* define a common agreed-upon terminology by providing concepts and relationships among the set of services from a real world domain. *Goals* are depictions of the expectations a service requestor may have when seeking for a service on the following aspects: functionality, approach and quality of service. *Mediators* coordinate the heterogeneity problem that occurs between descriptions at different levels [39]: data level - different terminologies, protocol level - different communication behaviour between services, and process level - different business processes. WSMO further defines four types of mediators: OO Mediators connect and mediate heterogeneous ontologies, GG Mediators connect Goals, WG Mediators link Web services to Goals, and WW Mediators connect Web

services resolving mismatches between them. *Web services* are descriptions of services that are requested by service requestors, provided by service providers and agreed between service providers and requesters.

In [19] we studied and compared existing semantic Web services frameworks used in various industrial and governmental sectors. We found the user perceptions and intentions to use either OWL-S or WSMO were bigger concerns than the expressiveness and reasoning capabilities of both facilitators. Hence our Figure 3 shows a more general framework of the semantics integration life cycle (SILC) for services development. Semantics and ontology are leveraged into each step in traditional Web service development and composition environments. The SILC adopts a top-down approach. It has several phases, including developing business processes (Business Process Modelling phase), adding technical and business constraints to processes (Development phase), annotating the composition workflow with a domain ontology to prepare semantically enriched service requests in a combined service flow (Enrichment phase), and deploying and executing in the final process (Runtime phase). Each phase of the SILC is responsible for performing a specific set of tasks. Instead of binding the combined services within the composition at design time (the development phase), selected services can be semantically described in the business processes. These semantic service requests can be annotated with the domain ontology. The domain ontology is managed within the scope of service management. The final business process is specified as a composite process in a workflow language enriched with process semantics such as OWL-S. The activities of preparing and sending a request for a Web service, discovering a service on the basis of matching requirements defined in QoS requirements or the service level agreement (SLA), and acquiring its response are dependent on semantic enhancements by the business participants, namely the service provider, the requester, and the registry of the whole service-oriented architecture. For example, for a mobile service, the location based service ontology will be a key part of the SLA, which is maintained in both the model repository and the semantics repository. The semantics repository, together with details of the SLA and QoS for those stand-alone services, will in turn be a core component of service management. Here we can see OWL-S is just one of many such efforts to provide ontological support for a services system.

3.2 QoS-Oriented Services Ontology Alignment

QoS is an important criterion for e-service selection in a dynamic environment. In general, QoS refers to the capability of a network to provide better service to selected network traffic over various technologies. For a decentralised system like a peer-to-peer (P2P) based network, the dynamic and unpredictable nature in e-service processes can significantly affect the service's composition and performance. In addition, the dynamic e-business vision calls for a seamless integration of business processes, applications, and e-services over the Web space and time. In other words, QoS properties such as reliability and availability for an e-service process are very important. Furthermore, changes and delay in traffic patterns, denial-of-service attacks and the effects of infrastructure failures, low performance in executions, and other quality issues over the Web create QoS complications in a P2P network. Quite often, unresolved QoS issues cause critical transactional applications to suffer from unacceptable performance degradation. Consequently, there is a need to distinguish e-services using a set of well-defined QoS criteria.

With the large number of e-services, it is desirable that consumers can distinguish between 'good' and 'bad' service providers. In such a case, QoS is the means to select a 'better' e-service among various providers. In addition, different collaborating e-services applications will compete for network resources in an unreasonable and uncontrollable manner if their interactions are not coordinated by agreements or specification on QoS differentiation. Naturally, these factors will force service providers to understand and achieve QoS-aware services to meet the demands. Also, a better QoS specification for e-service will become more significant by being a unique selling point for a service provider. Fundamentally, the Web services QoS requirement refers to both the functional and non-functional quality aspects of an e-service. This includes performance, reliability, integrity, accessibility, availability, interoperability and security [27]. The properties become even more complex when adding transactional features to e-services.

Notably, SILC is similar to traditional SDLC from the perspective of integrating legacy systems. As many legacy systems lack semantic or ontological artefacts, it is challenging to reengineer such systems to integrate these new components. However, both OWL-S and WSMO are successfully built upon the flexibility of multi-agent systems and the loosely-coupled feature of service oriented systems [19]. As we experimented in the proposed architecture and case study example (sections 4 and 5 respectively), we noted there would not be a huge amount of extra work to make invasive changes. Instead non-invasive componentised plug-play approaches would be sufficient to repurpose legacy systems to work with semantics enriched business processes.

3.3 Multi-agent system for services system

How to properly design and integrate QoS criteria in a decentralised e-service process is an important innovation for e-business development in a decentralised network. It particularly lends itself to ontology based development, as services correspond to tasks that can be indexed using an application ontology. In a dynamic environment, a higher domain ontology can be used by agents to locate appropriate providers of services and undertake dynamic evaluation through appropriate communication between agents. [18] presents an ontology-driven framework to build complex process models that can be reused in this application. More specifically, a web services modelling ontology is described in detail in [33] and a "Generic Negotiation Ontology (GNO)" in [12] as an upper level negotiation ontology for software agents. This existing research can be reused in this application.

The automatic support we propose consists of a Multi Agent System (MAS) providing a QoS evaluation to both service requestors and service providers. The system will identify service providers' capability and performance so as to enhance the service composition for service clients over the physical distributed service network. Due to the complexity of QoS metrics, a well-defined QoS service description does not actually exist. With a P2P architecture the QoS is gauged by a service client through cooperative interactions with other peers that can potentially provide the service. The scope of using ontologies in this web services development is available given that most of the current work focuses on the definition of QoS ontology, vocabulary or measurements and to a lesser extent on a uniform evaluation of qualities. Furthermore, a services unit of analysis nicely corresponds to a service carried by an agent.

When an agent receives a service request that it cannot fulfil, it seeks out a service from another agent or repository of services. This may happen as follows:

1. Identify the corresponding domain of the request using both domain and application ontology.
2. Use the domain knowledge to map to the service interface in order to index the task corresponding to the service requested as designated by the application ontology.
3. Map its application knowledge to the individual tasks and perform the tasks to fulfil the service request.

For example, suppose that an agent is interested in engaging in a specific negotiation with an opponent agent. Assuming it is aware of the negotiation protocol, with limited domain knowledge and information about its opponent's preferences, it needs a method to model the opponent and a method to devise a strategy to act. By mapping its application and domain knowledge to the services library, it identifies and employs a suitable coded implementation for a model and strategy. As the negotiation commences, the agent feeds information to the service model interface, the model updates, and the agent feeds the output of the model (along with the negotiation protocol) to the strategy interface, and follows the recommended course of action. The agent has no fixed automated negotiation approach but, rather, has the capacity to dynamically select the approach that best suits its circumstance. We will introduce the MAS system for the P2P based service composition in the next section.

4. The MAS based Architecture for P2P Service Composition

A pure P2P architecture lacks a proactive mechanism to incorporate good support for complex manipulation on the layers of ontology involved in a large scale business process oriented services system, as discussed in section 3. In this section a new architecture is introduced where an agent complements passive peers to select, compose and invoke available services, based on repositories of services and their related ontologies.

In such a new P2P agent-based architecture dedicated to sharing resources, the services system generally acts as an interface to a set of resources distributed across a network. Each agent within the system typically acts as a gate keeper to a local repository of resources that it shares with other similar peer agents as they broadcast their requests. In this architecture, all agents co-operate fulfilling queries and have access to their repository of resources whenever a query received can be assisted by their local resources. Resources shared can be information (files of data, music etc. e.g. as specified in systems similar to [22] and [30]) or services as in this paper and in [4]. In our P2P framework, the system consisting of all cooperating agents responds to requests by a user (e.g. a service requester, a software developer, a human web user) who is also represented by an agent in the P2P network that acts on his/her behalf. This agent aims to fulfill the request, e.g. locates services and responds to queries by other similar agents. The collection of all these agents and agents assisting them in their tasks form a P2P community-based cooperative system. For composing services using their semantics, a P2P agent-based system is shown in Figure 4. An agent (oblong in Figure 4) representing a user (hexagon in Figure 4) has access to a knowledge base containing services/resources that the user is willing to share with other users. Each service/file/resources (cylinder in Figure 4) is identified by a unique identifier within the P2P network (e.g. Service identifier, HTML, PDF, music or video).

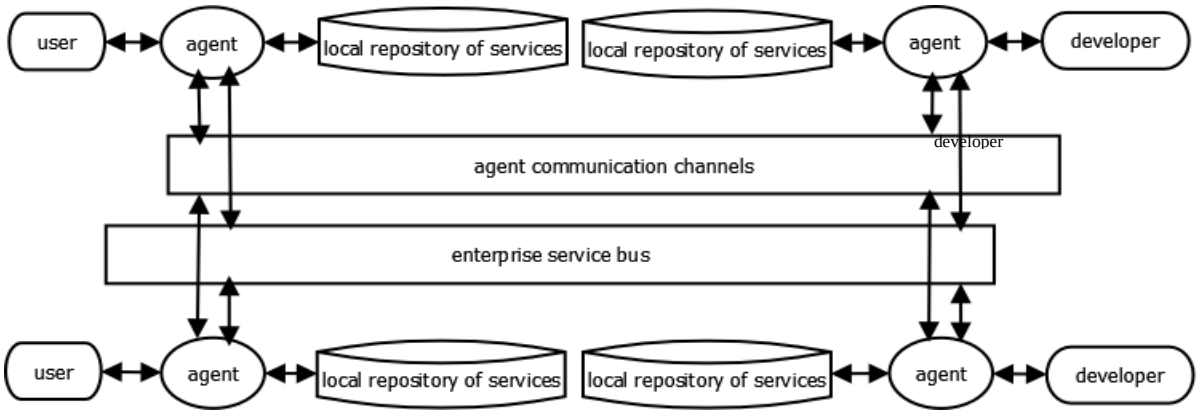

Figure 4. The P2P multi agent system

As agents automatically interact on behalf of users seeking services to be composed, communities of common interest emerge. These communities may overlap. Providers and users of services may belong to more than one community. For instance a service to '*open an account'* may belong to the community of banking developers as well as that for insurance developers. As more and more services are composed, agents become more efficient and effective by interacting with the agents in the communities most likely to be able to provide them with service components. The P2P system is responsible for locating sites where candidate services are available based on previous requests. The mediation between service requesters and providers is always done by the system. When an agent makes a service request, a candidate agent provider responds either by providing details about services it can supply or refusing the service. When all responses are received, the requesting agent combines and refines the results to compose a list of services that can be composed to fulfill the request. The requester agent then selects services it wants to compose and initiates the composition.

After a successful composition a requester's knowledge base is updated to include the received and the composed services. Similarly for all agents involved in processing a service request, their knowledge base is also updated with additional information reflecting the domain and attributes of the requester agent. This information is used in future service requests. Thus as agents interact they develop awareness of the services possessed by their peers and which peers may be interested in the services that they themselves offer. Each agent keeps a record of its history of service sharing in order to evaluate the QoS and to use this for future service requests. The collection of this history is in essence a distributed QoS ontology distributed across providers. The QoS ontology can provide assessments of past queries and providers, and also information to make QoS estimates for members. It is used to produce short lists of candidate nodes for future queries by calculating the similarity between the current query and a past query and its QoS. In a fully evolved P2P system, agents may use this QoS knowledge about other users' interests to request/negotiate information from their peers when they do not know who has services of interest. New providers are constantly added to the history, expanding the user-agent's contact circle.

The strategy of service sharing can be applied to any domain that can be prescribed by an ontology. The P2P service application system, which instantiate the service selection and execution, subsequently allows dynamic composition of Web services in a highly distributed and heterogeneous computing environment [4] as mentioned in section 2. Then at the architecture level, the system provides, to both service requestors and service providers, the QoS evaluation. The system will identify service providers' capability and performance so as to enhance the service composition for service clients over the distributed service network. Due to the complexity of QoS metrics [11, 31, 42], a well-defined full QoS service description does not yet exist. We propose to exploit the Web Service Modelling Ontology (WSMO) [33] as a complementary conceptual framework to create the QoS ontology to describe various perspectives on Web services, to facilitate integrating the services. In a specified domain, a problem solving method unit of analysis nicely corresponds to a service carried by an agent. The agents themselves can dynamically select service implementations that best suit the service or the QoS required matching the requested service level agreement (SLA). This selection will be made using a P2P searching mechanism to locate appropriate services from other peer agents. Cooperative communication between agents about their existing services, their past service requests and their performance will enable service requesters to locate the peer service provider with the most suitable QoS. The coordination and communications, which might occur among the involved agents and peers as well as service requesters and providers, would rely on the sharing and mapping between the different layers of ontologies.

The suggested architecture has been implemented in the enhanced UOW-SWS tool. Specifically the UOW-SWS tool has been extended to incorporate WSMO features to facilitate Web services selection via

ontological QoS and spatial information. UOW-SWS is a JXTA-based (jxta.dev.java.net) peer-to-peer workflow information system upgraded from SwinDeW-B [35].

5. Case Study: Ontology-Based Development of a Loan Approval Service System

In this section we use a typical loan approval service system to explain our ontology based development tool. It has been used by many e-commerce application prototypes, so we'd like to utilise it in UOW-SWS prototype to explain and demonstrate our selection method [44]. A loan application process deployed using our prototype. Initially, *Customer* sends a loan request to a financial organisation. A *Coordinator* peer, who has the knowledge of whole business process, will seek appropriate peers/agents to fulfil the task by sending *Pipe* messages and evaluating peers/agents' performance. For the whole process, it consists of two small single services (i.e., task or activity): "riskAssessment" and "loanApproval". "riskAssessment" provides the service (Assessor) about evaluating customer's reputation and loan amount, thus generating the risk assessment of the loan. Only when the risk assessment meets the requirement (e.g. good reputation and satisfactory permitted loan amount) of "loanApproval" (Approver), can the loan application be approved; otherwise, the loan request is rejected.

There have been many efforts to model services based business processes as graph based workflows, for example, [6] exploited Petri nets. As Figure 5 shows, a business process can be simply converted into the Control Flow Graph (CFG) form so as to ensure the coordinator splits the business process without constraints [44]. For the purpose of orchestrating and executing composite services, nodes in the CFG graph are basic activities. Each node knows a set of its predecessors and a set of successors as well as the conditions for it to be executed, if any. In this way, the task allocation for different peers can be intelligently performed. Therefore, the decentralised run-time environment can be effectively coordinated and self-managed with services located to wide area peer hosts who communicate with each other according to the de facto standard business process or workflow definitions.

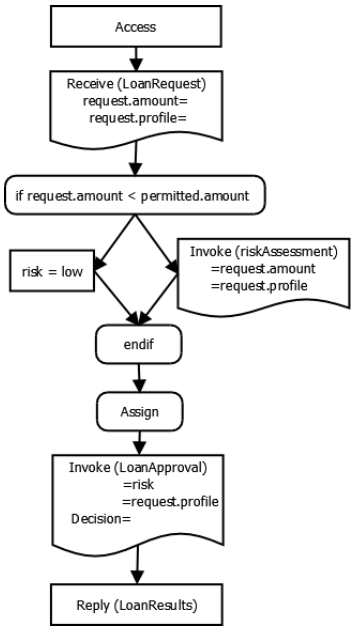

Figure 5. CFG in LoanApproval case

We can reuse an available banking ontology as domain ontology, and the typical tasks involved in a loan approval will form a real business application which is supported by a process ontology at the application layer. In our prototype, we implement the services as mobile web services so that the whole implementation is sufficiently complex to cover all layers of ontologies as discussed previously. The key to building such a system is how to implement a mobile service environment with the support of architectural ontologies and mobile agents or peers. Therefore, we first introduce a QoS ontology (using WSMO as a modelling ontology for services' agent and peer), and its subclass, a geographic location ontology (using OWL-S for profiles or WSMO again.). Later we will show the system setup and interfaces where objects defined in ontologies are instantiated.

There have been some projects studying QoS-empowered service selection. In [45], the authors present a QoS-aware middleware-supporting quality-driven Web service composition. Two service selection

approaches for constructing composite services have been proposed: local optimization and global planning. In [42], the authors focused on the creation of QoS ontology models which proposed QoS ontology frameworks aimed at formally describing arbitrary QoS parameters. From their on-going work, we are aware that they have not yet considered QoS-based service selection. The most similar work to ours, namely METEOR-S [25], bases the distribution of semantic Web service descriptions on a classification system expressed in service or registry ontologies. In our opinion, this solution is good in terms of globally organizing registries to benefit service management rather than for the service discovery or selection itself. Although it is relatively effective to publish and update service description information based on their categories, it would be difficult for service requestors to select services without understanding details of their principles. In contrast, our UOW-SWS tool is built by taking into consideration correlations between various service quality measurements and is based upon a well-founded peer-to-peer e-service workflow system.

5.1 QoS Ontology for Peer-to-Peer based Service System

Non-functional WSMO properties of the four WSMO elements typically describe non-functional aspects such as author, creation date and natural-language descriptions. In this paper, we introduced new QoS descriptors that could also be used in parallel with existing non-functional attributes of WSMO elements. These include of QoS, such as performance, availability and spatial features of distributed services. This QoS extension is similar to the common notion of non-functional properties in "*Web services*"; however, our QoS parameters are a more general non-functional properties. We develop the WSMO non-functional properties to support adaptive P2P-based service composition.

In our framework, *coordinator* roles are allocated to agents at runtime (detailed in Section 5.3). These organise the peer/agent selection process and distribute tasks. The resultant decentralised architecture is coordinated and self-managed with services allocated to peer/agent hosts who are able to communicate with each other according to a real business process agreement or standard workflow definitions. We now present a more effective representation to enable peers to evaluate a candidate composition and select the most appropriate peers for a requested service in a P2P information system. Based on [42], we define an extensible class QoSProperty that aims to extend nonFunctionalProperties class in WSMO for P2P-based service selection.

```
Class nonFunctionalProperties
     ...other existing properties...
     hasQoSProperty type QoSProperty
Class QoSProperty sub-Class nonFunctionalProperties
     hasPropertyName type string
     hasPropertyValue type {int, float, long, others}
     hasPreferredValueType type {low, high}
     hasWeight type float
```

Each QoS Property is generally described by *PropertyName* and *PropertyValue*. For the purpose of QoS-based selection, two additional properties are defined: *hasPreferredValueType* and *hasWeight.* The *hasPreferredValueType* property represents the desired trend. For example, the lower the response time is, the better the QoS that could be achieved. The *hasWeight* property is a value denoting the weight of the property, when considering several metrics. In this context, we define the weight value within the range [0, 1], while different end users may have different weight values for their service requirements. For instance, a peer's "ResponseTime" can be described in Web service profiles as following:

```
dc _"http://purl.org/dc/elements/1.1#",
webService _http://example.org/ LoanApprove
nonFunctionalProperties
  dc#title hasValue "Peer 1"
  dc#description hasValue "ResponseTime for LoanApprove process by peer 1"
  ……
  hasPropertyName hasValue _string ("ResponseTime")
  hasPropertyValue hasValue _int ("500")
  hasPreferredValueType hasValue _string ("low")
  hasWeight hasValue _float ("0.8")
endNonFunctionalProperties
```

From this example, we can see that QoS parameters for services, which support the banking domain and loan approval application, can be defined in this QoS ontology. They are prospectively mapped to the relevant

domain and application ontologies in which the system or application users may explicitly or implicitly define their requirements for underlying service systems. The effort for service selection and composition becomes a matchmaking process for peers and agents to find the best candidate of each service to achieve the overall QoS to meet the users' requirements. Hence the ontologies at different layers should be smoothly aligned.

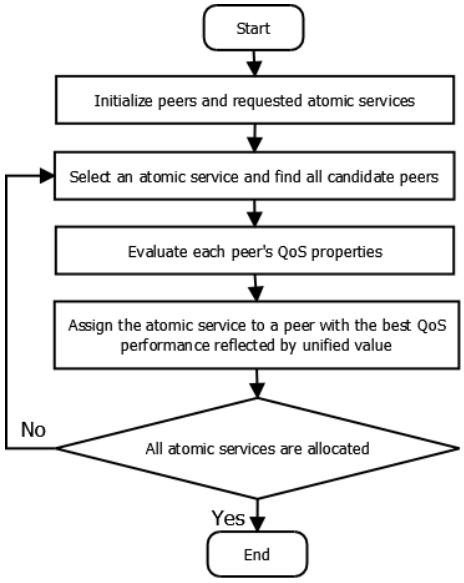

Figure 6. Flow diagramming for allocating atomic services to peers

To evaluate different non-functional properties of e-service peers, the central concepts in our modelling are: *PreferredValueType, Weight* and *Unified Value. PreferredValueType* has two possible values, "low" and "high". We utilise them to identify various non-functional properties. For example, "ResponseTime" is usually expected to be as short as possible when choosing an appropriate peer, so the PreferredValueType of "ResponseTime" is normally "low". Likewise, "ComputationCost" is also usually deemed "low" as it is unlikely anyone would prefer to choose a service with an expensive computation. However, "Reliability" and "Availability" are often considered as "high", since many systems have a high reliability and availability requirements. "*Weight*" indicates the importance and priority of certain properties during service composition so that the weight value varies from service to service and from property to property. Lastly, "*Unified Value"* gives each peer's overall quality measure which can be used to assess each peer's capability to meet the requirements of a requested service. To enable the peers' coordinating agent to intelligently select peers and plan a composition process, we sketch a selection process to assign the atomic services to appropriate peers within the service composition, as shown in Figure 6. This flow chart addresses the allocation method for multiple peer profile specifications and takes into account the above formulated objective.

5.2 An Instance of QoS Ontology: A Geographical Location Ontology

As an example of an ontology for non-functional QoS properties to support mobile services based bank loan approval application, we also develop a geographical ontology. In a decentralised network, geographical location is an important factor in both service selection and composition. In a P2P-based business process, a peer's geographic information is usually related to services' accessibility, particularly for a pervasive location based service, which integrates a mobile device's location with other information in order to provide added value to a context-aware user [20-21]. For example, with the popularity of smart phones and tablets such as iPhone and iPad, mobile banking is becoming more versatile. In many cases, agents or peers, that invoke the loan approver and loan assessor services, might be located at different work sites. Typically, there is no guarantee that a service can be selected or composed given the requested location requirements. In practical applications, business process managers must deal with alternatives that deviate from the requested service locations. We are interested in identifying those alternatives where the deviation is minimal, such as the nearest available service. In order to effectively enhance services' quality with respect to accessibility in a P2P network, we consider basic geographic information about a would-be task-allocated peer and incorporate it into the QoS profile as an extension of the previous QoS specification. To develop a geographical ontology which is generally a subclass of generic QoS ontology, we can use either WSMO or OWL-S constructs, i.e.,

profiles. Figure 7 shows the relations between these different ontology profiles in the context of OWL-S.

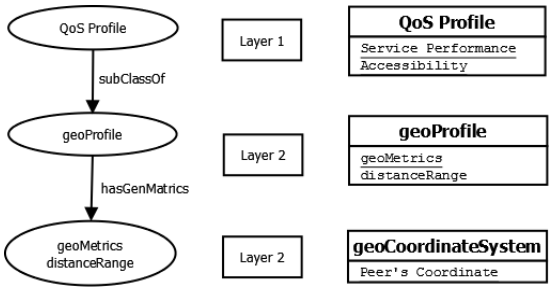

Figure 7. Relations between QoS profile and Geographical Profile for services

Figure 8 denotes the relationships between the QoS property and Geo property in WSMO, and the logic of using extensible geographic features for distributed services. In a real environment, it is not unusual that different applications may have different requirements, that is to say, some geographic properties (e.g. angle and region) do not need to be considered or evaluated for some applications (e.g. loan application case or simpler local services). Therefore, in order to be more flexible and practical, we define an attribute named "Essential" in GeoProperty, which means if "isEssential"=1, this kind of attribute is regarded as necessary and cannot be neglected.

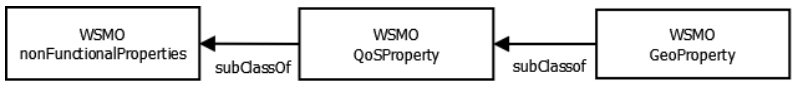

Figure 8. Extensible high-level relationships between properties

```
Class GeoProperty sub-Class QoSProperty
         hasGeoName type string
         hasGeoValue type {int, float, long, others}
         hasPreferedValueType type {low, high}
         hasWeight type float
         isEssential type boolean
```

The above is a definition of class "GeoProperty" which is the subclass of QoSProperty. In order to effectively enhance service quality for accessibility in a P2P network, we herein consider the geographic property about the peer and incorporate it into the QoSProperty Class as an extension of QoS specification. Here is an example for description of a peer's distance:

```
dc _"http://purl.org/dc/elements/1.1#",
webService _http://example.org/ LoanApprove
nonFunctionalProperties
  dc#title hasValue "Peer 1"
  dc#description hasValue "Distance between peer 1 and service requestor"
  ……
  hasGeoName hasValue _string ("Distance")
  hasGeoValue hasValue _int ("100")
  hasPreferedValueType hasValue _string ("low")
  hasWeight hasValue _float ("0.6")
  isEssential hasValue _boolean("1")
endNonFunctionalProperties
```

The metric of "Distance" is viewed as a basic criterion when choosing an appropriate peer to invoke a requested services, so it has been viewed as a necessary property in order to improve service accessibility. It is feasible to not only establish a better link between a potential service (or partially composed service) and an expected or requested set of location requirements, and also use this approach to rule out less viable compositions early in the process.

5.3 Implementation of the Mobile Loan Approval System

Technically, a coordinator in our prototype can adaptively organise the peer/agent selection process and distribute tasks. In this way, the decentralised run-time environment can be effectively coordinated and self-managed with services being located to wide area peer/agent hosts, who are able to communicate with each other according to a real business process agreement or standard workflow definitions. In this sub-section, we design a more effective and qualitative way for P2P information systems to distinguish which peer (or agent) is the most appropriate for a requested service.

As we mentioned in earlier sections, QoS and spatial perspectives involve many non-functional properties of e-services, such as: service availability, service accessibility, service performance and service geographic features. In a dynamic environment, the service selection with combined QoS and geographic specifications is often a quite complex process due to the diversity of metrics with different value types, value ranges and measurements. Taking account of correlations between those different specifications, we simplify and unify the various specifications to make the selection process less complicated and more effective. We may assign a set of random data and demonstrate the evaluation of four peers who are available in our loan case in using the UOW-SWS prototype, i.e., pre-deployed in the JXTA network. The properties of each peer use different metric units with correspondingly varying values. If the peer's properties are numerous, it would be extremely hard to distinguish and make an optimal decision as to which peer is the most appropriate for service composition.

We implemented a set of QoS based service selection algorithms (as indicated in Figure 6 – for details on these algorithms refer to [35, 44]) in our P2P based semantic Web service prototype. This selection method for the peers' combined specifications is reasonably suitable and effective to be fully adapted in a dynamic environment, especially in the sense of an autonomous way to select a best peer to perform any specific task at different stages.

To migrate UOW-SWS onto the Cloud platforms where existing vendor APIs are already available, we can easily establish a peer to cloud and peer (P2CP) model [37]. In this model Cloud users can download data and related ontological specifications from the storage cloud and exchange them with other peers, regardless of whether the other peers are cloud users or not. There are three data transmission tunnels in this storage model. The first is the cloud-user data transmission tunnel, which is responsible for data transactions between the cloud storage system and the cloud users. The second is the clients' data transmission tunnel, which is responsible for data transactions between individual cloud users. The third is the common data transmission tunnel which is responsible for data transactions between cloud users and non-cloud users.

Figure 9 (adapted from [37]) is an example to show how a P2CP model works. Therein Cloud user 2 is downloading data from data node 1, which is in the Cloud, via cloud-user data transmission tunnel. In the meantime, Cloud user 2 is exchanging data with Cloud user 1 and Cloud user 3 via the clients' data transmission tunnel, while common peers 2, 5, and 6 communicate via common data transmission tunnels. Other existing models such as Microsoft SharePoint tended to balance loads between peers and Cloud serves in different ways. However, in our P2CP model, peers may communicate directly and flexibly between each other without tight dependence on servers, though some advanced features such as backing up, caching, security and versioning of data may still be elevated or mitigated to servers just because the peers' storage and computing capacities are "supposedly" inferior to those of the Cloud servers.

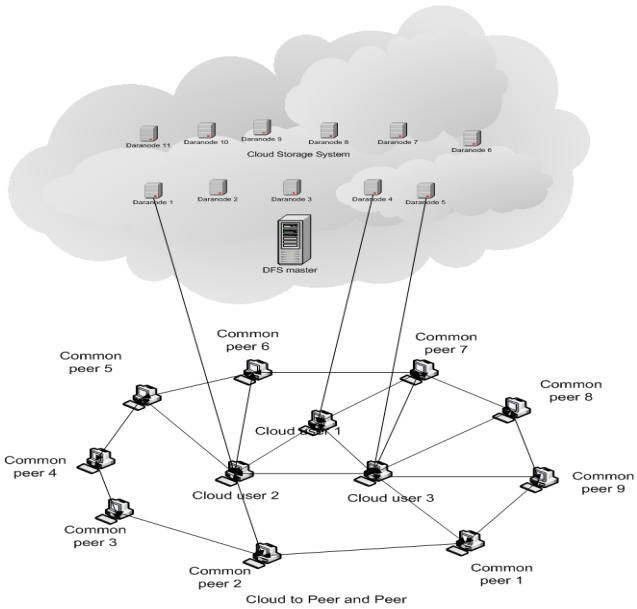

Figure 9. P2CP Network model.

6. Conclusions and Future Work

Using off-the-shelf domain ontologies as a starting point of system development is the focus of our efforts in the applied use of ontologies in our intelligent services development environment. This will enable the transfer and adaptation of existing techniques for ontologies, e.g. for mapping and translating between multiple ontologies to obtain a more economical approach to software engineering development and address interoperability and work product reuse. Not only will an ontology-based services framework be complete, consistent and produce systems that can easily be evolved to new contexts but it can have a highly developed maintenance phase to guide developers in reusing existing systems and components previously developed (using an ontological approach).

This paper promotes ontology-based software development with a focus on methodologies for services system development with the support of P2P architecture and multi-agent system. The paper first provided a 3 layer abstraction of ontologies to be applied in service oriented development lifecycle. Then the paper presented a semantic integration lifecycle for the development of a domain or application solution for business process problems, where architectural services ontology becomes essential and QoS turns out to be a major requirement. To make the aligned ontologies useful, a multi-agent system is needed to support the P2P e-services system. We implemented a tool to validate our approaches with a case study to demonstrate how the concepts and constructs proposed in this paper work together.

Much work remains to refine the concepts presented in this paper and to ensure that they are applicable to areas where the use of ontologies is less obvious than the banking domain discussed in this paper. Towards this, the first step is to develop required ontological techniques. These include ontology-based techniques for consistency checking across products and processes, and ontology-based techniques for testing completeness of products and processes within and across methodologies. Underlying complex issues need to be resolved, e.g. as how to reconcile requirements from multiple sources and multiple versions of ontologies from different domains, applications and service based architectures. Another issue is how candidate problem solving methods get identified to be reused. It may well turn out that an ontology-based approach is most suited to areas of applications where the set of possible agent actions are well specified in advance e.g. in modelling service oriented systems. Modern large scale service oriented systems, such as Google Play, mobile applications or social networks, might put more focus on the performance improvement of the overall system, in which ontology enrichment should not be an added burden.

References

[1]. *Ahmed, W., Aslam, M.A., Lopez-Lorca, A.A, Shen, J., Beydoun, G. and Richards, D. (2011), Using ontologies to synchronize change in relational database systems, Journal of Research and Practice in Information Technology, 43(2): 89-108.*

[2]. *Berners-Lee, T. (2005). Web for real people (Keynote address). 14th International World Wide Web Conference (WWW05). Japan.*

[3]. *Beydoun, G., Gonzalez-Perez, C., Henderson-Sellers, B. et al. (2006). Developing and Evaluating a Generic Metamodel for MAS Work Products. In Software Engineering for Multi-Agent Systems IV: Research Issues and Practical Applications. A. Garcia, R. Choren, C. Lucenaet al. Berlin, Springer-Verlag. LNCS 3914, pp.126-142.*

[4]. *Beydoun, G., García-Sánchez, F., Vincent-Torres, C.M, Lopez-Lorca, A.A., and Martínez-Béjar, R. (2013), Providing metrics and automatic enhancement for hierarchical taxonomies, Information Processing and Management, 49(1): 67-82.*

[5]. *Beydoun, G., Lopez-Lorca, A.A., García-Sánchez, F. and Martínez-Béjar, R. (2011), How do we measure and improve the quality of a hierarchical ontology?, Journal of Systems and Software, 84(12): 2363-2373*

[6]. *Cicirelli, F., Furfaro, A., Nigro, L. (2010) A service-based architecture for dynamically reconfigurable workflows, Journal of Systems and Software, 83(7): 1148-1164*

[7]. *Cordi, V., Mascardi, V. et al. (2004). Developing an Ontology for the Retrieval of XML Documents: A Comparative Evaluation of Existing Methodologies. In 6th International Workshop on Agent-Oriented Information Systems (AOIS2004) at CaiSE'04, pp.73-87.*

[8]. *DARPA. (2007). DAML Ontology Library. Retrieved 28 May, 2013, from http://www.daml.org/ontologies/.*

[9]. *Demirkan, H., Kauffman, R. J., Vayghan, J. A,. Fill, H. Karagiannis, D. and Maglio, P. P. (2008). Service-oriented technology and management. Journal of Electronic Commerce and Applications 7(4): 356-376.*

[10]. *Dileo, J., Jacobs, T. et al. (2002). Integrating Ontologies into Multi-Agent Systems Engineering. 4th International Bi-Conference Workshop on Agent Oriented Information Systems (AOIS2002). Italy, pp.15-16.*

[11]. *Dong, F., Luo, J., Song, A., Cao, J. and Shen, J. (2012), An effective data aggregation based adaptive long term CPU load predictions mechanism on computational grid, Future Generation Computer Systems: the international journal of grid computing: theory, methods and applications, 28(7):1030-1044*

[12]. *Ermolayev, V. and Keberle, N. (2006): A Generic ontology of rational negotiation, in Proceedings of Information Systems Technology and its Applications, 5th International Conference ISTA, Germany, pp.51-66*

[13]. *Forte, M., Lopes de Souza, D., Francisco do Prado, A. (2008) Using ontologies and Web services for content adaptation in Ubiquitous Computing, Journal of Systems and Software, 81(3): 368-381*

[14]. *Gartner. (2008a). "27 Technologies in the 2008 Hype Cycle for Emerging Technologies." 2008 Press Releases Retrieved 1/10, 2011.*

[15]. *Gartner. (2008b). "Key Predictions for IT Organisations and Users in 2008 and beyond." 2008 Press Releases Retrieved 1/10, 2011.*

[16]. *Girardi, R. and Serra, I. (2004). Using ontologies for the specification of domain-specific languages in multi-agent domain engineering. CAiSE Workshops (2), pp.295-308*

[17]. *Goth, G. (2009). Reaping Deep Web Rewards Is a Matter of Semantics. Internet Computing 13(3): 7-10.*

[18]. *Greco, G., Guzzo, A., Pontieri, L. and Saccà, D. (2004) An Ontology-Driven Process Modeling Framework, in 15th International Conference on Database and Expert Systems Applications, pp.13-23*

[19]. *Kamaruddin, L. A., Shen, J. and Beydoun, G. (2012). Evaluating usage of WSMO and OWL-S in semantic web services. Proceedings of the Eighth Asia-Pacific Conference on Conceptual Modelling (APCCM 2012) (pp. 53-58). Melbourne: Australian Computer Society*

[20]. *Kapitsaki,G.M., Prezerakos, G.N., Tselikas, N.D., and Venieris, I.S. (2009) Context-aware service engineering: A survey, Journal of Systems and Software, 82(8): 1285-1297*

[21]. *Kim, J. W., Kim, J. Y. and Kim, C.S. (2006) Semantic LBS: Ontological Approach for Enhancing Interoperability in Location Based Services. On the Move to Meaningful Internet Systems: OTM 2006 Workshops, LNCS, Vol.4277, pp.792-801.*

[22]. *Klampanos, I.A. and Jose, J. M. (2003) , An Architecture for Peer-to-Peer Information Retrieval, in SIGIR'03, pp.401-402*

[23]. *Kohlborn, T., Korthaus, A., Chan, T. and Rosemann, M. (2009). Identification and Analysis of Business and Software Services: A Consolidated Approach. IEEE Transactions on Services Computing, 2(1): 50-64.*

[24]. *Lassila, O. and Hendler, J. (2007). Embracing Web 3.0. Internet Computing 11(3): 90-93.*

[25]. *LSDIS: METEOR-S: Semantic Web services and Processes. Available at: http://lsdis.cs.uga.edu/projects/ meteor-s*

[26]. *Madougou, S., Shahand, S., Santcroos, M., van Schaik, B., Benabdelkader, A., van Kampen, A. and Olabarriaga, S. (2013), Characterizing workflow-based activity on a production e-infrastructure using provenance data, Future Generation Computer Systems, 29(8): 1931-1942.*

[27]. *Mani, A. and Nagarajan, A. (2002): Understanding quality of service for Web services, http://www.ibm.com/developerworks/library/ws-quality.html. accessed in June, 2013*

[28]. *Martin, D. and Domingue, J. (2007a). Semantic Web Services, Part 1. Intelligent Systems, IEEE 22(5): 12-17.*

[29]. *Martin, D. and Domingue, J. (2007b). Semantic Web Services, Part 2. Intelligent Systems, IEEE 22(6): 8-15.*

[30]. *Mine, T., Matsuno, D., Kogo, A., and Amamiya, M. (2004) Design and Implementation of Agent Community Based Peer-to-Peer Information Retrieval Method, in Cooperative Information Agents (CIA2004). 2004. Germany: Springer-Verlag, pp.31-46*

[31]. *Negri, A., Poggi, A,. Tomaiuolo, M. and Turci, P. (2006) Ontologies and web services: Agents for e-business applications. In Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS '06), pp.907-914, ACM Press.*

[32]. *Pilioura, T. and Tsalgatidou, A. (2009). Unified publication and discovery of semantic Web services. ACM Trans. On Web 3(3): 1-44.*
[33]. *Roman, D., Keller, U., Lausen, H., Bruijn, J. De, Lara, R.., Stollberg, M., Polleres, A,. Feier, C., Bussler, C. and Fensel, D. (2005): Web Service Modeling Ontology, Applied Ontology, 1(1), 77-106*
[34]. *Shen, J., Grossmann, G. et al. (2007). Analysis of business process integration in Web services contexts. Future Generation Computer Systems 23(3): 283-294.*
[35]. *Shen, J., Yuan, S., and Krishna, A. (2009), Dynamic selection of service peers with multiple property specifications, in The 6th International Conference on Business Process Management (BPM 2008) Workshops, Lecture Notes in Business Information Processing, vol. 17, pp.609-620.*
[36]. *Shi, X. (2008). The Challenge of Semantic Web Services. Intelligent Systems, IEEE 23(2): 5-5.*
[37]. *Sun, Z. and Shen, J. (2013) A high performance peer to cloud and peer model augmented with hierarchical secure communications, Journal of Systems and Software, 86(7): 1790-1796*
[38]. *Sousa, J.P.P., Carrapatoso, E., Fonesca, B,. Pimentel, M.G.C and Bulcao-Neto, R.F. (2009). Composition of Context-Aware Mobile Services Using a Semantic Context Model, IARIA International Journal on Advances in Software, 2(2): 1-13*
[39]. *Toma, I., Foxvog, D.and Jaeger, M.C. (2006). Modeling QoS characteristics in WSMO. Workshop on Middleware for Service Oriented Computing (MW4SOC'06), pp.42-47.*
[40]. *Tran, QNN, Low, GC and Beydoun, G. (2006). A Methodological Framework for Ontology Centric Agent Oriented Software Engineering. International Journal of Computer Systems Science and Engineering, 21, pp. 117-132.*
[41]. *Tran, Q.N.N., Beydoun, G. and, Low, G. (2007). Design of a peer-to-peer information sharing MAS using MOBMAS (ontology-centric agent oriented methodology. In Advances in Information Systems Development, Springer pp. 63-76.*
[42]. *Tsesmetzis, D.T., Roussaki, I.G., Papaioannou, I.V and Anagnostou, M.E. (2006) QoS awareness support in Web-Service semantics, Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT/ICIW 2006), pp.128.*
[43]. *Yuan, X., Min, G., Ding, Y., Liu, Q., Liu, J., Yin, H. and Fang, Q. (2013), Adaptive resource management for P2P live streaming systems, Future Generation Computer Systems, 29(6): 1573-1582.*
[44]. *Yuan, S. and Shen, J. (2008). Mining E-Services in P2P-based Workflow Enactments, Journal of Online Information Review, Emerald Group Publishing.. 32 (2): 163-178.*
[45]. *Zeng,L., Benatallah,B., Ngu,A., Dumas, M., Kalagnaanam J. and Chang, H. (2004), QoS-aware middleware for Web services composition. IEEE Transaction on Software Engineering 30 (5): 311 – 327*