



Shen, J., Yang, Y., Yan, J. (2007). A p2p based service flow system with advanced ontology-based service profiles

Originally published in *Advanced Engineering Informatics*, 21(2): 221-229

Available from:

<http://dx.doi.org/10.1016/j.aei.2006.05.001>

Copyright © 2007 Elsevier Ltd.

This is the author's version of the work. It is posted here with the permission of the publisher for your personal use. No further distribution is permitted.



A P2P based Service Flow System with Advanced Ontology-Based Service Profiles*

Jun Shen^{1,2}, Yun Yang², Jun Yan^{1,2}

1-School of Information Technology and Computer Science

University of Wollongong

Wollongong, NSW, Australia 2522

2-Faculty of Information and Communication Technologies

Swinburne University of Technology

Hawthorn, Melbourne, VIC, Australia 3122

emails: jshen@uow.edu.au, yyang@ict.swin.edu.au, jyan@uow.edu.au

Abstract

A peer-to-peer (p2p) based service flow management system, SwinDeW-S, could support decentralised Web service composition, deployment and enactment. However, traditional workflow definition languages, such as extended XPDL and service-oriented BPEL4WS, have become insufficient to specify business process semantics, especially the descriptions of inputs, outputs, preconditions and effects. In this paper, we propose a novel solution based on OWL-S, a semantic Web ontology language that leverages service discovery, invocation and negotiation more effectively. The enhanced SwinDeW-S architecture is adapted with advanced ontology-based service profiles, and it takes advantage of a well-developed profile generation tool, which translates the BPEL4WS process models to the OWL-S profiles. As a result, in a new prototype equipped with both BPEL4WS and OWL-S, communications and coordination among service flow peers have become better organised and more efficient.

Keywords: Web services, p2p workflows, business processes, service profiles, ontology

1. Introduction

A wide range of workflow topics have been largely addressed for about two decades [8, 9, 23]. Traditionally, workflow management systems adopt the client-server based centralised architecture to manage the enactment of processes. However, centralised coordination has exhibited vulnerability, inflexibility and human restriction, which have been witnessed as major obstacles to wide deployment of workflow systems in the real world [22]. To deal with these problems, we have argued that centralised architecture is poorly mismatched with the inherently decentralised nature of workflow [21], and presented a p2p [3] based decentralised workflow system SwinDeW

* This paper is an extended version of an earlier paper appeared in Proceedings of the 9th International Conference on Computer Supported Cooperative Work in Design (CSCWD'05), Coventry, UK, May 2005. The research work reported in this paper is jointly supported by Australian Research Council under Discovery Project Grant DP0663841 and Linkage Project Grant LP0562500.

(*Swinburne Decentralised Workflow*) [21, 22]. To improve system openness, especially to explicitly describe IOPE (inputs, outputs, preconditions and effects) of workflow activities and tasks in light of service specification standards like WSDL [6], we have upgraded SwinDeW to SwinDeW-S (*SwinDeW* for Services) to support deployment and enactment of Web services for carrying out concrete processes [15]. With SwinDeW-S, existing Web services can be orchestrated in a naturally decentralised p2p environment more flexibly. Once services are deployed, coordinated and monitored workflow peers from anywhere may invoke or execute them automatically. Primarily, the heterogeneity involved may be tackled by XML-based business process and service languages, so the prototype can be applied in general enterprise integration environments.

However, today's Web services do not allow defining the business process semantics of Web services. Thus, Web services are inherently isolated from one another. Breaking isolation means connecting Web services and specifying how a collection of Web services are jointly used to realise more complex functionalities [18, 24]. Nowadays, there are many XML based workflow process definition and execution languages (process modelling languages) like BPEL4WS (Business Process Execution Language for Web services) [5], XPDL (XML Process Description Language) [19], etc. that can be used to describe business processes in the service flow systems like SwinDeW-S. They are specialised languages for describing most aspects of the workflow and representing the business process logic. The Meta model of process modelling languages varies dramatically from one to another [1]. The interoperability issues will affect the effective data sharing and exchange between the p2p-based workflow or coordination systems [14, 20]. Distinguished among many semantic Web service projects, OWL-S (Ontology Web Language for Services) is an attempt to provide an ontology for describing Web services profiles [11]. OWL-S has well-defined semantics based on OWL, making it computer interpretable and unambiguous. It also enables the definition of Web services content vocabulary in terms of objects and complex relationships between them, including class, subclass, and cardinality restrictions.

The distinctive research reported in this paper is carried out in the context of SwinDeW-S, which aims at providing genuinely decentralised workflow support based on the p2p technology in a BPEL4WS compatible Web service environment. The major motivation and focus of this paper are to comprehensively discuss the critical issues of service profile support in SwinDeW-S to take advantages of OWL-S based service ontology methods and tools, including our prototypes which create and communicate OWL-S profiles among service flow peers.

The rest of this paper is organised as follows. The next section describes the background and identifies the requirements. In Section 3, we give an extended SwinDeW-S architecture based on OWL-S, followed by details of the build-up process of service profiles from BPEL4WS. A couple of prototype tools will be introduced in Section 4. Section 5 discusses the related work and Section 6 concludes the contribution and outlines future work.

2. Background and Requirements Analysis

2.1 SwinDeW-S: Extending SwinDeW with Web Services

The novel framework of SwinDeW neither implies the presence of a centralised data repository for information storage, nor a centralised workflow engine for coordination. The basic working unit, known as a peer, is denoted as a software component residing on a physical machine, which operates on behalf of an associated workflow participant. Each peer represents one or more capabilities (roles) in workflow processes. Given essential information and authority, a peer is a self-managing, autonomous entity that is able to communicate with other peers directly to carry out workflow functions. The internal components and data repositories of each peer, as well as the interactions among them, are detailed in [21, 22].

In order to be autonomous, each peer requires essential process information in accordance with its capabilities. Thus, SwinDeW presents a distinct data storage philosophy known as “*know what you should know*” [21, 22]. Unlike conventional approaches where each participant either knows nothing or all about the workflow processes, each peer only gains relevant knowledge about the processes. SwinDeW partitions a process into individual tasks, then each task definition is distributed to relevant peers appropriately according to a capability match, i.e., the definition of a task requiring a certain capability is distributed to peers with this specific capability. Therefore, peers in the system have partial but essential knowledge which enables them to collaborate in order to fulfil all the key functions of the complete workflow execution including process instantiation, work allocation, instance navigation and execution monitoring.

With regard to run-time functions, SwinDeW mainly focuses on the issues of process instantiation and instance execution. The phase of process instantiation creates various task instances and assigns them to various performers. Peers coordinate with one another directly to create various task instances at dispersed locations. A process instance is eventually represented by a network of peers performing various tasks in certain orders. Instead of static work allocation, the task instance is assigned through the direct and automatic negotiation by

relevant peers, taking workload balance and performance optimisation into consideration. Correspondingly, the execution of a process instance does not rely on a centralised service to perform coordination. The work is passed from one peer to another for execution, through direct communication. The SwinDeW decentralised workflow system helps to take advantage of p2p computing and thus yields a competitive advantage. A JXTA-based (JXTA is a Java based P2P development kit) prototype has been implemented for demonstration and evaluation purposes. The core services of SwinDeW, including the peer management service, the process definition service, the process enactment service and the monitoring and administration service, are realised through the invocation of the JXTA core services.

However, SwinDeW is mainly developed for workflow coordination. Support for real world workflows needs to be added. With Sun's AppServer and J2EE key elements like JAX-RPC, we can further deploy peers' processes as end-points of diverse Web services and execute them through SOAP calls [15]. Document transfer between peers also becomes flexible with SAAJ (SOAP with Attachments API for Java). Each peer must be deployed and registered with some services, as described in the WSDL interfaces, in order to create and run a task. Each peer can only create and run tasks that require the services that this peer can provide. If service deployment has not been done already, a list of services needs to be created so that all peers can choose from the same list. For this to work properly, all the peers in the community need to be running when a new capability is added so that they can all receive the information. Certain service discovery mechanisms similar to UDDI but based on peer capabilities are also complementary.

Initially, SwinDeW-S uses an extended XPDL process definition language to describe activities. XPDL belongs to the family of graph-structured process definition languages. The Meta model of XPDL utilises the concept of activity (workflow process activity) as the core component of the process definition (workflow process definition) [19]. The workflow process definition is built from individual workflow process activities that are related to form a control flow via transitions, which are governed by transition information. XPDL describes a workflow process definition in terms of what is to be done, when it has to be done, under what conditions, and by whom or what. Suppose for a scenario in a PurchaseOrder system, the peer responsible for the order will take buyer's request sheets as input and return the product prices, we need to exploit non-standard extended attributes to define IOPE as follows:

```

<ExtendedAttributes>
  <ExtendedAttribute Name="input" Value="ProductRequestSheet"/>
  <ExtendedAttribute Name="output" Value="ProductPriceList"/>
  <ExtendedAttribute Name="description" Value="1"/>
  <ExtendedAttribute Name="duration" Value="1"/>
  <ExtendedAttribute Name="resources" Value=""/>
  <ExtendedAttribute Name="tool" Value=""/>
  <ExtendedAttribute Name="capability" Value="PurchaseOrder"/>
  <ExtendedAttribute Name="timestamp" Value="1101945011746"/>
</ExtendedAttributes>

```

To better integrate XPDL inputs and outputs with WSDL/BPEL messages, and to better describe preconditions and effects in an explicit syntax and clear semantics for peers, some new workflow based service languages should be investigated to support the service profile descriptions. Based on our work in integrating BPEL4WS and OWL-S [18], we are motivated to choose advantageous OWL-S as a service profile description language.

2.2 BPEL4WS, OWL-S and Service Profiles

Now we look at the industry standard language, BPEL4WS. A BPEL4WS process definition provides or uses one or more WSDL services, and provides the description of the behaviour and interactions of a process instance related to its partners and resources through Web service interfaces. A BPEL4WS process is a kind of flow-chart of activities, which can be either primitive or structured. The set of primitive activities contains: <invoke>, <receive>, <reply>, <wait>, <assign>, <throw> and <empty>. Several structured activities are defined to enable the presentation of complex structures. These are <sequence>, <switch>, <pick>, <flow>, <compensate>, <scope> and <while>. Every BPEL4WS process is exposed as a Web service using WSDL that describes the entry and exit points of the process. Meanwhile, WSDL data types are used to describe the information being passed within the process, and WSDL might be used to reference external services required by the process [5]. As BPEL4WS has gradually become the basis of a standard for Web service composition.

However, BPEL4WS has several shortcomings that limit the ability to provide a foundation for seamless interoperability [10, 13]. For example, BPEL4WS uses WSDL port type information for service descriptions, but WSDL does not describe the side-effects or preconditions of the services, and the expressiveness of service behaviour and inputs/outputs is restricted to the interaction specification; BPEL4WS does not represent inheritance and the relationships of an individual service amongst others; BPEL4WS does not provide well-defined semantics for automated composition and execution. Failure to fulfil these requirements is due to

BPEL4WS's reliance on XML-based service descriptions. XML provides a rudimentary content language, but lacks the constructs to specify complex relationships between Web resources. Therefore, researchers are investigating how formal semantics models can be exploited to enhance specification capabilities of workflow languages such as BPEL4WS [1, 16]. Comparably, with the popularity of the semantic Web, ontology defines a common vocabulary for stakeholders who need to share information in a domain. It includes machine-interpretable definitions of basic concepts in the domain and relations among them. The use of ontology provides a very powerful way to describe objects and their relationships. Using ontology in our context will overcome the XML limitations. Some milestone languages used in the deployment of ontology are described in [16].

OWL-S is an ontology language for Web services, with the support of language constructs from OWL. The ontology of services can be divided into three parts, namely, service profile, service model and service grounding [11]. The OWL-S service profile describes what is required from users or agents (peers), and accordingly what a service can provide to them. The service model describes the service's process model, i.e. the control flow and data flow involved in using the service. The OWL-S service model defines three types of processes, namely, atomic, simple and composite processes. It is the most translatable part to the BPEL4WS process model. The OWL-S service grounding defines how to bind communication level protocols and message descriptions in WSDL. All OWL-S components are annotated with classes of well-defined types that make the service descriptions machine-readable and unambiguous. Additionally, the ontological structure of types allows type definitions to draw properties from the hierarchical inheritance and relationships to other types.

In a service profile and service model, the top level class of OWL-S process ontology is *process*. A process may have any number of IOPE. A process may also involve any number of participants. Atomic processes can be directly invoked as they have no sub-processes and can be executed in a single step. Composite processes can be decomposed into other processes, which are linked by control constructs such as *if-then-else*. In contrast to atomic processes, simple processes cannot be directly invoked, but like atomic processes, they are viewed as a single-step abstract execution.

To enhance SwinDeW-S with these new workflow and business process ontology languages, development of a new architecture that integrates the multi-level process specification and profiles is crucial. Moreover, the integration of the BPEL4WS process model and OWL-S service model becomes essential to better adopt advanced ontology based service profiles to a p2p process coordination environment.

3. An Extended SwinDeW-S Architecture with Profile Support

3.1 SwinDeW-S with OWL-S

In the original SwinDeW-S, the lowest layer is JXTA, which supports basic p2p functions that will be exploited by the middle layer peers, which are hosted by various business process systems. The upper layer is composed of deployed services, which are supported by Web service interfaces, like WSDL. At the process instantiation stage, a peer with the service capability that matches a task request is selected for the task. When we use BPEL4WS or XPDL to describe the composition of services to make up a complex Web service, it is easy for a centralised engine to directly interpret a BPEL4WS process description and deploy it. However, it is difficult for a p2p based service engine, because each peer does not and should not know all the information about the process, but only the information that is necessary to carry out its mission. Therefore, it is necessary to appropriately convey the information presented by the structured activity in a p2p network.

We can convert a BPEL4WS process into a control flow graph (CFG) form. Nodes in a CFG graph are basic activities. Each node is constrained by a set of its predecessors and a set of successors as well as the conditions for it to be executed. Together with their own knowledge, the nodes maintain the same IOPE information kept by the structured activities. The knowledge about links, which form certain structures, is also preserved in each node. The tasks assigned to specific peers will be based on the partition of nodes and related IOPE profiles. However, the peer discovery requires the description of activities to be available, accessible and explicable to the peer, who hosts the BPEL4WS process description, and at the same time to the other peers, which are capable of performing the specific activities. At run-time, a temporary coordinator peer resolves a BPEL4WS process into basic activities. This coordinator will discover a suitable peer to host the service by sending a request message to peer groups that are able to perform an activity. Upon receiving the message, peers in each peer group will negotiate to find the most suitable peer for the activity or service. The chosen peer will notify the ad-hoc coordinator with a reply message. When all activities have known their associated peers, the coordinator will generate routing data so that each peer knows where its predecessors and successors are, where the activities at the other end of its source links are and where the activities that use the same variables are, according to IOPE matchmakings, which are mainly based on WSDL descriptions. After the routing data are set up the coordinator will send the detailed

service profile of each activity to corresponding peers. After these coordination and negotiation stages, the process can be executed through a set of SOAP invocations.

The more semantics the process description and service profile specifies, the more possibility for the most suitable peer to be selected for an activity, hence the more effective interoperability the system can offer. Nowadays, OWL-S has drawn wider and wider attention from different research communities. Many software tools of editors, reasoners and deployers are available to make its integration onto existing Web services prototypes and products easier and more flexible. With OWL-S's semantics support, decentralised workflows like SwinDeW-S, which were traditionally specified by XPDL or BPEL4WS, could support well-formed coordination between peers who are autonomously enacting activities in business processes. Figure 1 shows the extended SwinDeW-S. Within this new architecture, peers communicate among themselves through interfaces supporting both BPEL4WS and OWL-S profiles, regardless of the types of role they are playing. In other words, such integration allows a peer to play complementary roles.

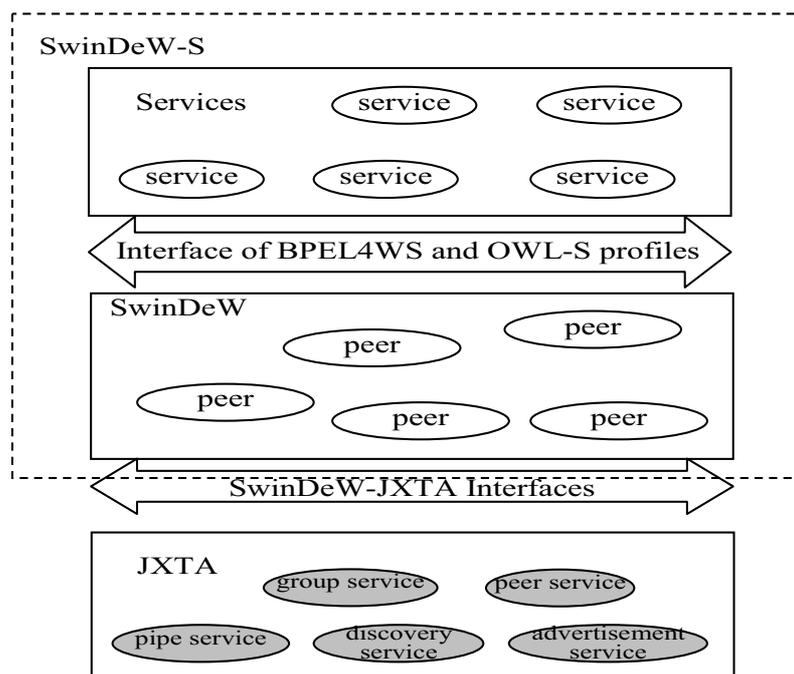


Figure 1: SwinDeW-S with Service Profile Support

3.2 Establishments of Service Profiles from BPEL4WS

To enable executable service invocation and enactment in the new SwinDeW-S architecture, it is necessary to connect the BPEL4WS workflow model to the OWL-S service profile model. When a BPEL4WS based

specification of a business process is available, we need not re-invent or re-compose the related OWL-S profiles from scratch. On the contrary, we can generate templates of service profiles directly by mapping from BPEL4WS specifications. This feasibility is attributable to the similarity of specified business process structures between BPEL4WS and OWL-S. In this section, we will briefly outline how constructs of BPEL4WS can be mapped onto their OWL-S equivalents. The complete mapping specification and examples can be found in [18].

3.2.1 Processes and variables in a service and data flow profile

A BPEL4WS executable process can be directly invoked. Hence it will be mapped onto an OWL-S atomic or composite process, depending on the internal activity of the executable process. OWL-S atomic processes can be directly invoked and composite processes can be made explicitly invocable by setting the *invocable* property of the composite process to TRUE. Nevertheless, BPEL4WS abstract processes, which cannot be directly invoked, represent the abstract view of the processes and hence will be mapped onto OWL-S simple processes. An OWL-S simple process can be thought of as an abstract view on either an atomic or a composite process [11].

Corresponding to BPEL4WS descriptions, the WSDL message element is used to represent the abstract definition of the data being transmitted in and out of the processes [6]. A message element consists of one or more logical parts. Each part is associated with a type (DTD or XSD). Since we know the data types of parts of the messages by the *type* attribute, all the messages will be represented as an OWL class and the type of the class will not be restricted to any particular data type but OWL object (i.e. *Thing*). A part element of a message will be mapped onto the property element of the corresponding OWL class and the data type of the property will be based on the type specified for that part in WSDL. Since the message element in WSDL uses the XSD type system to associate the data types [6] with its parts, we may have customised and built-in data types defined in a XML schema. To use the data types defined in the external schema declarations and to associate them with our data types, we need to import these schema declarations in our process ontology. We will annotate the namespaces in the process OWL-S profile.

Variables in BPEL4WS store and transfer the messages which constitute the state of a business process. The messages held by workflow engines are often received from the partners or to be sent to the partners via primitive activities. The type of each variable may be a type of the message, an XML schema simple type or an XML schema element. The *messageType*, *type* or *element* attributes are used to specify the type of a variable. Variables

that are defined by the *messageType* attribute represents a message thus sharing the same data type as the message it refers to. Such types of variables are mapped onto the message data types in the relevant OWL-S process ontology. The inputs and outputs variables and messages of the atomic processes derived from <receive>, <reply> and <invoke> activities, will be described in a data flow OWL file, which uses the OWL-S *valueOf* class to refer to their respective super-class atomic process's inputs and outputs. When composing atomic processes into composite processes (i.e. when a composite process has atomic sub-processes in it), it is crucial that the inputs and outputs of the sub-processes are related to each other. This is addressed using the OWL-S *valueOf* class (representing that two parameters used for referencing are equal), similar to referring the inputs and outputs of the derived atomic processes to their corresponding super-class atomic process's inputs and outputs. Composite processes can be handled similarly, but variables in primitive and structured BPEL4WS activities will represent the complete data flow for both atomic and composite respectively. From the dataflow point of view, the relationships between the atomic classes beneath the process can be implicitly indicated. Once the description of the composite process is complete, the atomic processes can be described in linear order while their inputs and outputs can be linked accordingly.

3.2.2 Profiles of activities

All the primitive activities in BPEL4WS are mapped to the atomic process in OWL-S but there are some slight differences among them. Although they occur in BPEL4WS it is WSDL which actually defines them initially in its *operation* part which resides in *portType*. The <reply> activity allows the business process to send a message only. So it only has an output. We can find this basic activity appearing quite often in BPEL4WS which is equivalent to <output> in a WSDL *operation*. We can get this <reply> information either in BPEL4WS or in WSDL *portType*. However the former provides more information about this activity in the whole business process while the latter only keeps the basic message content. In OWL-S, we map it to the atomic process with the property *hasOutput*. <receive> is quite similar to <reply> and the only difference is that <receive> merely has inputs. Due to <invoke>'s own characteristics, its mapping is the combination of <reply> and <receive>. The <throw> activity is used to generate an exception message in the business process. Because OWL-S has not specified how to handle the fault currently, we temporarily treat <throw> exactly the same as <reply> with the attribute *faultVariable* corresponding to the *outputVariable* within <reply>.

As the structural activity is composed of more than one activity that can be primitive activities or structural activities or both, we translate the structural activities to the composite processes in OWL-S accordingly. Composite processes play the same role in OWL-S like the function of structural activities in BPEL4WS. A composite process contains more than one sub process (atomic or composite process). OWL-S has a minimal set of control constructs. The basic primitives are *Sequence*, *Split*, *Split + Join*, *Unordered*, *Condition*, *If-Then-Else*, *Repeat-While*, *Repeat-Until*.

The <sequence> activity contains one or more activities that are executed sequentially. The activities are executed in the order that they appear within the <sequence> element. When the final activity in <sequence> has been completed, the <sequence> activity itself is completed. We express the <sequence> activity as an OWL-S composite process with control construct *sequence*. Because there is no directly corresponding control construct in OWL-S to model the logic of the <switch> activity, we use multiple *If-Then-Else* to fulfil this logic. Furthermore, in OWL-S, we use control construct *Repeat-While* to model the <while> activity with *whileCondition* by copying attribute *condition* in the <while> activity. The <pick> activity will run one of its event handlers within itself when this handler is activated. We map it onto the composite process with the *Choice* control construct from the original and explicit format. This treatment can stipulate all the actions that the <pick> activity has, but cannot specify the situation which has been clearly handled in BPEL4WS by using *onMessage*, *onAlarm* and other event handlers. The expression of the <flow> activity in OWL-S is the composite process with the *composedOf* and *Split* control construct. However *Split* only specifies concurrency and we are still waiting for further specifications to deal with waiting and synchronisation.

3.3 Example of Service Profiles in SwinDeW-S

We herein re-visit the PurchaseOrder example mentioned at the end of Section 2.1, in a WSDL/BPEL4WS environment, input and output messages will be defined in related WSDL files accordingly as shown below:

```
<portType name="PurchaseOrder">
  <operation name="check">
    <input message="def:ProductRequestSheet"/>
    <output message="def:ProductPriceList"/>
  </operation>
</portType>
```

If equipped with OWL-S service profile support, more explicit syntax and semantics will be incorporated as follows:

```

<process:AtomicProcess rdf:ID="order">
  <process:hasInput>
    <process:Input rdf:ID="ProductRequestSheet1">
      <process:parameterType rdf:resource="#ProductRequestSheet"/>
    </process:Input>
  </process:hasInput>
  <process:hasOutput>
    <process:UnConditionalOutput rdf:ID="ProductPriceList1">
      <process:parameterType rdf:resource="#ProductPriceList"/>
    </process:UnConditionalOutput>
  </process:hasOutput>
</process:AtomicProcess>

```

Currently, as automatic service discovery and dynamic binding of component services at run-time are important aspects in orchestrating composite services, more and more significant attempts have been made to leverage OWL-S so that rich semantics can be added to Web service descriptions. In SwinDeW-S, the corresponding peers join or leave peer groups by claiming or disclaiming the service capabilities which they have at the deployment or execution stage. The flexibility of this grouping process can also be boosted with explicitly accompanied OWL-S specifications, which can even support the descriptions of service quality metrics (for example, QoS parameters [14]). With OWL-S evolving, SwinDeW-S is open to integrate more advanced Web service descriptions and profiles that are equipped with rich semantics.

4. Prototypes

For service profiles, we had developed a tool (BPEL4WS2OWL-S) to support the mapping of business processes defined in BPEL4WS onto OWL-S based process ontology [18]. In this prototype, after a BPEL4WS file is inputted, WSDL files are distinguished in terms of a master WSDL file and some slave WSDL files. The master WSDL file is the essential one, which is connected to the BPEL4WS file and referred by all slave WSDL files. BPEL4WS explains the business process activities specifically and accurately. WSDL describes all the data used in a business process. Accordingly, we built our process structure in OWL-S closely relevant to BPEL4WS but in a totally different way. BPEL4WS uses flow to describe the business process, while OWL-S converts the process to a hierarchical structure. The tool allows easy navigation and representation of the hierarchical inheritance and relationships between the objects and their properties. Users can simultaneously view the tree structure representing the object view of the files (BPEL4WS as inputs and OWL-S as outputs) and also the source in Internet browsers.

The generated OWL files from our tool can be imported into Protégé (<http://protege.stanford.edu/>), an ontology editor. The derived subsumption and interaction relationships between activities in the business processes can be explicitly identified through OWL classes, instances, properties and slots. The messages and constraints that describe IOPE are also correlated to each other. The screenshot depicted in Figures 2 illustrates instances, properties and slots that appeared in the PurchaseOrder example process specification (as described in Section 3.3).

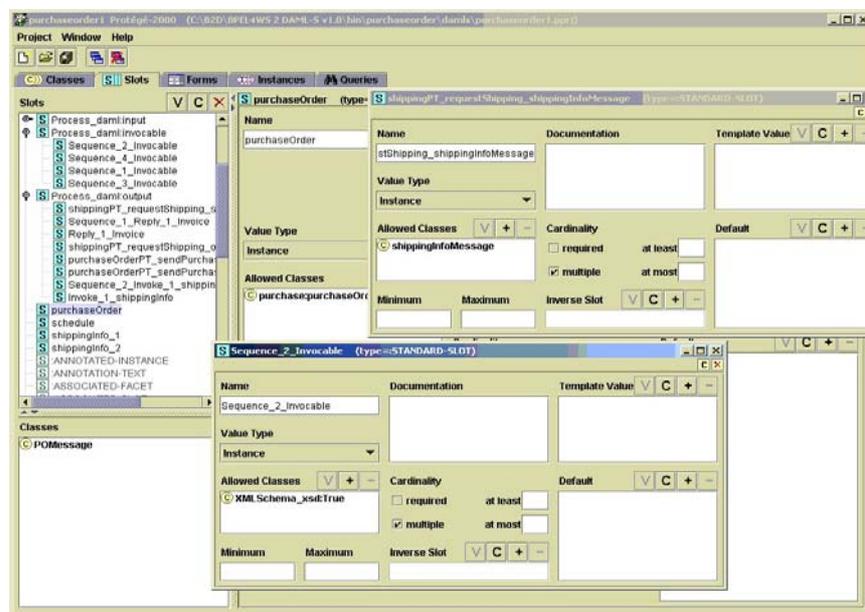


Figure 2: Slots of PurchaseOrder Process

On the other hand, we have implemented a new prototype called SwinDeW-B to better support BPEL4WS and OWL-S [17], besides the p2p-based SwinDeW(-S) as discussed earlier and reported in [15, 22]. In Figure 3, peer 4 (monitored through the right bottom window) acts as a coordinator while the other three peers are enacted to perform appointed tasks in a business process. At first, the coordinator peer analyses the BPEL4WS specification of the whole process and identifies the primitive activities within. Then it will notify the other available peers through broadcasted advertisements and will wait for these peers to respond. The interested peers will match their own WSDL interface and OWL-S profile, either self-specified or translated from BPEL/WSDL definition, with IOPE of the requested activities. If a peer can make a successful match and is not committed to other tasks, it will respond to the coordinating peer with an acknowledgement in due time. Once the coordinator peer has confirmed that all activities have been assigned to specific peers, it will initiate the process and enact the tasks of those

peers, who will accordingly invoke corresponding services to finish the tasks. The inputs and outputs between tasks are communicated as service I/O messages among peers.

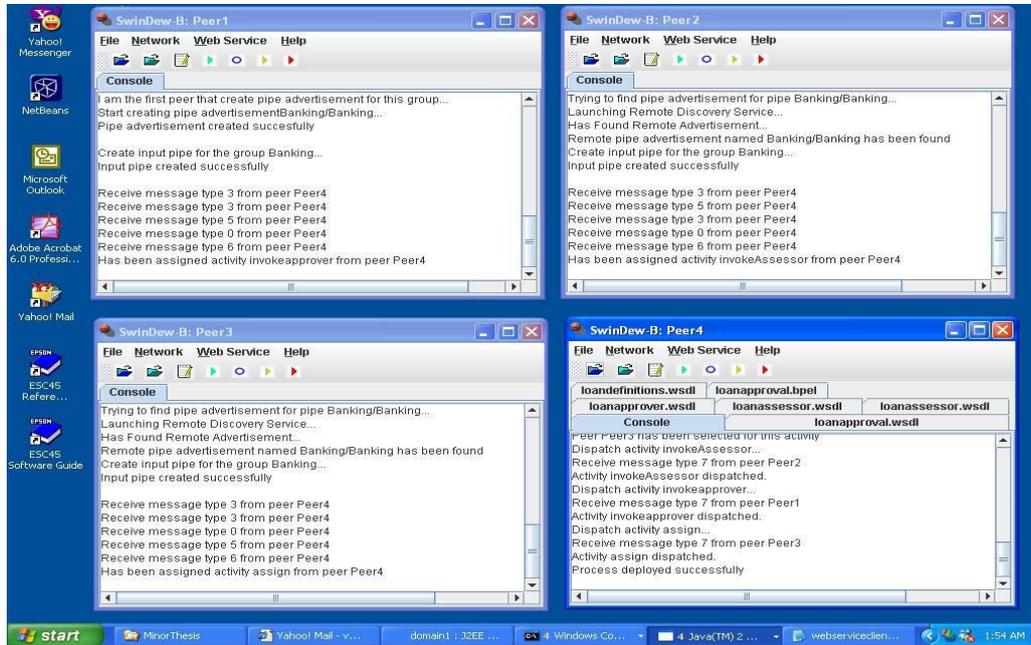


Figure 3: Peers Coordination with Service Profile

5. Related Work

From a system perspective, a number of researchers have revealed that centralised workflow management based on the client-server architecture has faced more and more challenges in supporting decentralised workflow applications [4, 21]. There is a growing trend that the next generation of workflow systems will be built in a truly decentralised manner. More specifically, p2p-based workflow systems have been recognised as one of the most strategic future directions for workflow research [12, 20]. Serendipity-II [8] is a decentralised process management environment developed at University of Waikato and University of Auckland, New Zealand. It supports collaborative and distributed process modelling and enactment through point-to-point connections for distributed software development projects. Another p2p-based workflow project conducted at Manchester Metropolitan University presents a p2p architecture for dynamic workflow management, which is based on concepts such as Web Workflow Peers Directory (WWPD) and Web Workflow Peer (WWP) [7]. Developed at San Diego Supercomputer Centre, the Matrix (www.npaci.edu/DICE/SRB/matrix) project delivers Grid workflow protocols and workflow language descriptions necessary to build a p2p infrastructure for Grid workflow management systems. This middleware allows applications and services based on Web service standards to

communicate with data and other resources in a Grid environment. Based on the rationale that workflow's decentralised nature needs to be supported more naturally, SwinDeW-S aims at providing p2p-based, genuinely decentralised workflow support in order to offer unique advantages such as better performance, increased reliability and scalability, enhanced user support, and improved system openness with Web service deployment and enactment [15, 22]. Furthermore, we have taken the primary consideration of advanced profile support for p2p based service systems by integrating BPEL4WS and OWL-S [18].

On the other hand, some research has been done on augmenting semantics to workflow languages and Web services. YAWL [1] is a workflow language based on the workflow patterns [2]. Its formal semantics allows YAWL to be used for the purposes of expressiveness and interoperability. Meteor-S [14] aims to extend Web services standards with semantic Web technologies to achieve greater dynamism and scalability. Research groups at Stanford University and Carnegie Mellon University have led the work in adapting BPEL4WS or WSDL for semantic Web or OWL-S [10, 13] to support ontology based service profiles. Until now, such work only maps WSDL service descriptions to the OWL-S service profile, which only deals with IOPE of related processes. They promise that the automatic service composition and discovery can be realised by matchmaker inference mechanisms. In WSDL2DAML-S [13], only the mapping of port types and operations to their corresponding DAML-S atomic processes is presented. It does not reflect the mapping of the message element in DMAL-S process ontology. We extended the mapping of WSDL2DAML-S by including the message element in our OWL-S based workflow process ontology. Our work reported in [18] is a significant complementary work to the above mentioned efforts. It is also a good stand-alone support tool for SwinDeW-S in service profile creation by integrating BPEL4WS, WSDL and OWL-S together.

6. Conclusions and Future Work

In this paper we have discussed service profile issues in the context of SwinDeW-S, a service flow extension of the peer-to-peer based decentralised workflow system, SwinDeW. Although service based workflow languages, such as BPEL4WS and related WSDL, could be integrated into the system, they still lack enough support for description of IOPE (inputs, outputs, preconditions and effects) if deployed along with traditional extended XPDL-based service profile descriptions. Therefore, a new ontology language for Web services, OWL-S, has been integrated and deployed into the extended prototype.

OWL-S enables definitions of the Web services content vocabulary in terms of objects and complex relationships between them including classes, sub-classes, cardinality restrictions, and hierarchical inheritance. It also provides a shared set of terms describing the application domain with a common understanding for sharing information and knowledge and with well-defined semantics. Using OWL-S as the ontology in our project, we not only aim to tackle the limitations and weaknesses of BPEL4WS and XML, but also try to solve the data integration and interoperability problems and issues faced today in the Web services world.

Within extended SwinDeW-S, especially our new tools that integrate BPEL4WS/WSDL and OWL-S, communication and coordination among peers can be enhanced from the semantics point of view. Some advanced features, like challenging descriptions of QoS awareness, will be developed in the future.

Reference

- [1] W. van der Aalst, A. H. M. ter Hofstede, YAWL: Yet Another Workflow Language, *Information Systems*, 30(4): 245-275, 2005.
- [2] W. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski and A. Barros, Workflow Patterns, *Distributed and Parallel Databases*, 14(1): 5-51, 2003.
- [3] K. Aberer and M. Hauswirth, Peer-to-Peer Information Systems: Concepts and Models, State-of-the-art, and Future Systems, *Proc. of the 8th European Software Engineering Conference (ESEC) and 9th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE-9)*, 326-327, Vienna, Austria, Sept. 2001.
- [4] G. Alonso, C. Mohan, R. Guenthoer, D. Agrawal, A. El Abbadi and M. Kamath, Exotica/FMQM: A Persistent Message-Based Architecture for Distributed Workflow Management, *Proc. of IFIP WG8.1 Working Conference on Information Systems for Decentralized Organizations*, Trondheim, August 1995. Also available as IBM Research Report RJ9912, IBM Almaden Research Center, Nov. 1994.
- [5] T. Andrews, F. Curbera, H. Dholakia, et.al, *Specification: Business Process Execution Language for Web Services*, Version 1.1, <http://www-106.ibm.com/developerworks/Webser vices/library/ws-bpel>, 2003.
- [6] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, *Web Services Description Language (WSDL) 1.1*, <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>, 2001.
- [7] G. J. Fakas and B. Karakostas, A Peer to Peer Architecture for Dynamic Workflow Management, *Information and Software Technology Journal*, 46(6): 423-431, 2004.
- [8] J. Grundy, M. Apperley, J. Hosking, and W. Mugridge, A Decentralised Architecture for Software Process Modelling and Enactment, *IEEE Internet Computing*, 2(5):53-62, Sept/Oct. 1998.
- [9] S. Liu, A. Goh and E. Soong, A Formal Framework to Support Workflow Adaptation, *International Journal of Software Engineering and Knowledge Engineering*, 12(3): 245-268, June, 2002.
- [10] D. J. Mandell and S. McIlraith. Adapting BPEL4WS for the Semantic Web: The Bottom-Up Approach to Web service Interoperation, *Proc. of 2nd International Semantic Web Conference (ISWC'03)*, LNCS 2870, 227-241, 2003.
- [11] D. Martin et al (eds) *OWL-S 1.2, OWL-based Ontology for Services*, <http://www.daml.org/services/owl-s/>, Mar. 2006.
- [12] J. B. Masters, Peep-to-Peer Technologies and Collaborative Work Management: The implications of "Napster" for

- Document Management, in L. Fischer, ed. *Workflow Handbook 2002*, 81-94, 2002.
- [13] M. Paolucci, N. Srinivasan, K. Sycara, and T. Nishimura, Toward a Semantic Choreography of Web services: From WSDL to DAML-S, *Proc. of 1st International Conference on Web Services (ICWS'03)*, 22-26, 2003.
- [14] A. Patil, S. Oundhakar, A. Sheth and K. Verma, Meteor-S: Web Service Annotation Framework, *Proc. of the 13th International Conference on World Wide Web (WWW'04)*, 553-562, 2004.
- [15] J. Shen, J. Yan, Y. Yang, SwinDeW-S: Extending P2P Workflow Systems for Adaptive Composite Web Services, to appear in *Proc. of Australian Software Engineering Conference (ASWEC'06)*, 2006.
- [16] J. Shen and Y. Yang, Extending RDF in Distributed Knowledge-Intensive Applications, *Future Generation Computer Systems*, 20(1): 27-46, 2004.
- [17] J. Shen, Y. Yang and Q. H. Vu, SwinDeW-B: A P2P Based Composite Service Execution System with BPEL, *Proc. of Workshop on Dynamic Web Processes (DWP 2005) within the 3rd International Conference on Service Oriented Computing (ICSOC)*, available as IBM RC23822, 73-84, 2005.
- [18] J. Shen, Y. Yang, C. Zhu and C. Wan, From BPEL4WS to OWL-S: Integrating E-Business Process Descriptions, *Proc. of 2nd IEEE Service Computing Conference (SCC'05)*, 181-188, 2005.
- [19] WfMC, *Workflow Process Definition Interface-XML Process Definition Language*, Version 1.0, Lighthouse Point, FL, http://www.wfmc.org/standards/docs/TC-1025_10_xpdl_102502.pdf, 2002.
- [20] G. Xexéo, A. Vivacqua, J. M. de Souza, et.al., COE: A Collaborative Ontology Editor based on a Peer-to-Peer Framework, *Advanced Engineering Informatics*, 19(2): 113-121, 2005.
- [21] J. Yan, *A Framework and Coordination Technologies for Peer-to-peer based Decentralised Workflow Systems*, PhD Thesis, Swinburne University of Technology, Australia, 2004.
- [22] J. Yan, Y. Yang and G. K. Raikundalia, SwinDeW - A Peer-to-peer based Decentralised Workflow Management System, to appear, accepted by *IEEE Transactions on Systems, Man and Cybernetics, Part A* in January 2005, available at <http://www.ict.swin.edu.au/personal/yayang/papers/SwinDeW-TSMC.pdf>.
- [23] Y. Yang, Enabling Cost-Effective Light-Weight Disconnected Workflow for Web-based Teamwork Support, *Journal of Applied Systems Studies*, Cambridge, England, 3(2): 437-453, 2002.
- [24] M. Younas, K-M. Chao and C. Laing, Composition of Mismatched Web Services in Distributed Service Oriented Design Activities, *Advanced Engineering Informatics*, 19(2): 143-153, 2005.