

Adapting P2P based Decentralised Workflow System SwinDeW-S with Web Service Profile Support

Jun Shen¹, Yun Yang¹, Jun Yan^{1,2}

¹ CICEC - Centre for Internet Computing and E-Commerce,
Faculty of Information and Communication Technologies, Swinburne University of Technology
PO Box 218, Hawthorn, Melbourne, Australia 3122
[jshen, yyang]@it.swin.edu.au

² CIAMAS - Centre for Intelligent Agent and Multi-Agent Systems,
Faculty of Information and Communication Technologies, Swinburne University of Technology
PO Box 218, Hawthorn, Melbourne, Australia 3122
jyan@it.swin.edu.au

Abstract

SwinDeW, a novel peer-to-peer based decentralised workflow, has been upgraded to SwinDeW-S, which supports Web service deployments and enactments to enhance system openness. To specify business process semantics, descriptions of input, output, precondition and effects, traditional workflow definition languages, such as extended XPDL which was used in SwinDeW, have become insufficient. Even new service-oriented business process languages, such as BPEL4WS, are unable to support the full description of service profile either. In this paper, we propose a new framework based on OWL-S, a semantic Web ontology language that leverages service discovery, invocation and coordination more effectively.

Keywords: Peer-to-peer Workflows, Web Services, Business Processes, Service Profile, Workflow Coordination.

1. Introduction

The Web services paradigm is poised to become a dominant form of distributed computing this decade and beyond. Web services involve a family of related XML-based protocols to describe, deliver, and interact with services. The most well-known protocols are Simple Object Access Protocol (SOAP), Web services Description Language (WSDL) and Universal Description, Discovery and Integration (UDDI). WSDL is the most important one in our context. WSDL files include a set of standard elements. These elements describe a particular Web service, including its interfaces and usage [4]. In the meantime, workflow management systems have become the most promising solutions for the organisations that need to automate their business processes in Web service environments. Applying workflow to a business process brings the

details of that process into focus and adds the required business rules and logic in the process.

A wide range of workflow topics have been largely addressed for about two decades [6, 7]. 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 the major problems for wide deployment of workflow systems in the real world. To deal with these problems, we have argued that centralised architecture is poorly mismatched with the inherently decentralised nature of workflow [15], and presented a peer-to-peer (p2p) [1] based decentralised workflow system SwinDeW (Swinburne Decentralised Workflow) [15-19]. To improve system openness, we have upgraded SwinDeW to SwinDeW-S (*SwinDeW* for Web Services) to support deployment and enactment of Web services, which carry out concrete processes.

However, today's Web services do not allow defining the business process semantics of Web services, thus, they are isolated. Breaking isolation means connecting Web services and specifying how collections of Web services are jointly used to realise more complex functionality [13]. Nowadays, there are many XML based workflow process definition and execution languages (process modeling languages) like BPEL4WS (Business Process Execution Language for Web services) [3], XPDL (XML Process Description Language) [14], etc. that can be used to describe workflow systems and business processes in the Web services world. They are specialised languages for describing all aspects of the workflow and representing the business process logic. The meta models of each language vary dramatically from one specification to another. The integration of Web services into higher levels of abstractions or frameworks is critical otherwise it will lead to interoperability issues that should be considered for effective data sharing and exchange

between the p2p-based workflow systems [12]. OWL-S (Web Ontology Language for Services, formerly DARPA Agent Markup Language for services, DAML-S) is an attempt to provide ontology for describing Web services profiles [9]. OWL-S has a 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 distinct research reported in this paper was carried out in the context of SwinDeW-S, which aims at providing genuinely decentralised workflow support based on the p2p technology in a Web service environment. The major focus of this paper is to deeply discuss the critical issues of service profile support in SwinDeW-S.

The rest of this paper is organised as follows. The next section briefly introduces the SwinDeW workflow system, including its decentralised system design and corresponding mechanisms. Then in Section 3, we present SwinDeW-S and its current problems. Issues of service profile support are identified by comparing and integrating XPDL, BPEL and OWL-S descriptions, and some initial solution framework is given in Section 4. Finally, section 5 discusses related work and section 6 concludes the contribution and outlines future work.

2. Background: SwinDeW

The novel framework of SwinDeW implies the presence of neither 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 which 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 [15]

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*" [16]. Unlike conventional approaches where each participant either knows nothing or all about the workflow processes, each peer only gains relevant knowledge about processes. In detail, SwinDeW partitions a process into individual tasks. After that, 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 capability. Therefore, peers in the system have partial and essential knowledge, which enables them to collaborate in order to fulfil all

the key functions of workflow execution, including process instantiation, work allocation, instance navigation and execution monitoring.

Regarding run-time functions, SwinDeW mainly focuses on the issues of process instantiation and instance execution [17, 18]. The phase of process instantiation creates various task instances and assigns 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.

Moreover, SwinDeW has been extended to offer incompletely specified process support [19]. A multi-level process modelling and execution approach is proposed which naturally fits into the SwinDeW overall framework. Run-time articulation work is modelled as essential workflow tasks towards final goals. Thus, on-the-fly process specification can be realised by coordinating the execution of these special tasks.

The SwinDeW decentralised workflow system helps to take advantages of p2p computing and thus yield competitive advantages. A JXTA-based prototype has been implemented for demonstration and evaluation purposes. The results so far are promising.

3. SwinDeW-S: Extending SwinDeW with Web Services

3.1 Introduction of SwinDeW-S prototype

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 essentially for workflow coordination. Support for real world workflows needs to be added. Therefore we upgrade SwinDeW to SwinDeW-S.

With Sun's AppServer and J2EE key elements like JAX-RPC, we can further deploy peers' processes as end-points (EP) of diverse Web services and execute them through SOAP calls. Document transfer between peers also becomes flexible with SAAJ. Figure 1 shows the architecture of SwinDeW-S.

Each peer must have some services in order to create and run a task. Each peer can only create and run tasks that require the services that particular peer can provide. If this 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 need to be

running when you add a new capability so that they all can receive the new information.

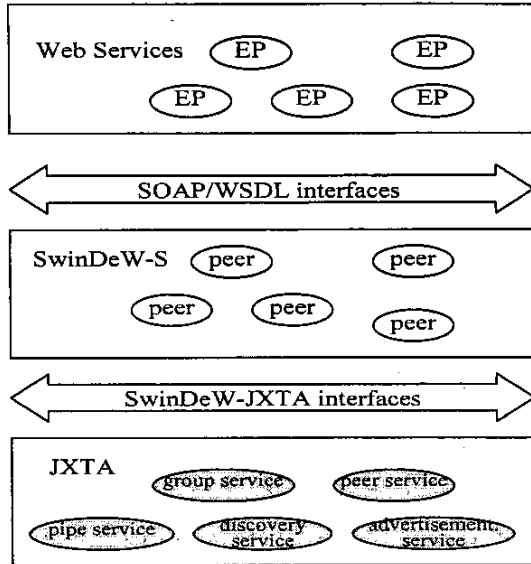


Fig.1 Architecture of SwinDeW-S

3.2 Problems with SwinDeW-S

Currently, SwinDeW-S still uses an extended XPDL process definition language to describe activities. Suppose a scenario in an online examination system. The automatic checking peer will take exam-takers' answer sheets as input and return the scores. We need to exploit non-standard extended attributes to define IOPE (input, output, precondition and effects) as follows:

```
<ExtendedAttributes>
  <ExtendedAttribute Name="input"
    Value="AnswerSheet"/>
  <ExtendedAttribute Name="output"
    Value="Score"/>
  <ExtendedAttribute Name="description"
    Value="1"/>
  <ExtendedAttribute Name="duration"
    Value="1"/>
  <ExtendedAttribute Name="resources"
    Value="" />
  <ExtendedAttribute Name="tool"
    Value="" />
  <ExtendedAttribute Name="Capability"
    Value="App exam"/>
  <ExtendedAttribute Name="timestamp"
    Value="1101945011746"/>
</ExtendedAttributes>
```

However, to better integrate with WSDL messages as input and output, 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.

4. Service Profile Support with OWL-S for SwinDeW-S

4.1 OWL-S and Service Profile

Now we look at a new industry standard language, BPEL4WS. A BPEL4WS process definition provides and/or uses one or more WSDL services, and provides the description of the behaviour and interactions of a process instance relative to its partners and resources through Web service interfaces. BPEL4WS leverages WSDL in three ways: (1) every BPEL4WS process is exposed as a Web service using WSDL that describes the entry and exit points for the process; (2) WSDL data types are used to describe the information being passed within the process; and (3) WSDL might be used to reference external services required by the process [3]. BPEL4WS supports the implementation of any kind of business process in a very natural manner and has gradually become the basis of a standard for Web service description and composition. However, it has several shortcomings that limit the ability to provide a foundation for seamless interoperability. The semantics of BPEL4WS is not always clear, thus complicating the adoption of the language. Major limitations in BPEL4WS specification have been listed in [8, 11]. Herein we may reiterate a few: BPEL4WS uses WSDL port type information for service descriptions; WSDL does not describe side-effects or preconditions of services, and the expressiveness of service behaviour and inputs/outputs is restricted to the interaction specification; BPEL4WS does not represent inheritance and relationships of an individual service and among the services; BPEL4WS does not provide well-defined semantics for automated composition and execution.

OWL-S is an attempt to provide ontology for Web services, within the framework of OWL. The ontology of services can be divided into three parts, which are characterised by the kind of knowledge provided about a service [9]. The service profile describes what the service requires from users or agents and what it provides to them. The service model describes the service's process model (the control flow and data-flow involved in using the service). OWL-S service model defines three types of processes (atomic, simple and composite). It is designed to enable automated composition and execution of services and is related most closely to the BPEL4WS process model. Service grounding connects the process model description to communication level protocols and message descriptions in WSDL. These 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.

The top level class of OWL-S process ontology is *process*. A process can have any number of inputs and outputs, preconditions, effects and participants. There

are three disjoint subclasses of the *process* class. Atomic processes can be directly invoked, have no sub-processes and execute in a single step (from the perspective of the service requester). Composite processes can be decomposed into other (atomic or composite) processes, which are linked by control constructs such as *sequence* or *if-then-else*. In contrast to atomic processes, they cannot be directly invoked. And simple processes cannot be directly invoked, but, like atomic processes, they are viewed as having single-step executions.

4.2 An extended SwinDeW-S framework

Without OWL-S semantics' support, neither XPDL nor BPEL4WS could organise well-formed negotiation and coordination between peers, who enacted real tasks and activities. Therefore, we propose a framework as shown in Figure 2, where peers communicate with each other through not only BPEL4WS interactions but also OWL-S profiles, no matter they are playing which type of role, for example, as a coordinator. Thus, the integration of BPEL4WS and OWL-S introduced in this paper becomes more essential for them to play complementary roles (see section 4.3 for details).

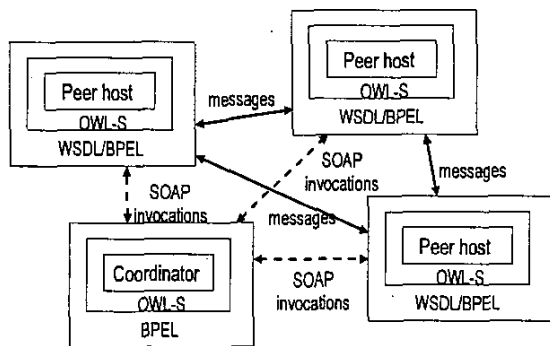


Fig.2 SwinDeW-S with service profile support

Nowadays, OWL-S has drawn wider and wider attention from different research communities, a lot of editors, reasoners and deployers software are available to make its integration onto existing Web services prototype and products easier and more flexible.

4.3 Integration of BPEL4WS and OWL-S

To enable executable service invocation and enactment, it is an important requirement to extend the BPEL4WS workflow model to the OWL-S service model. We had developed a tool (BPEL4WS2OWL-S v1.0)¹ to support the mapping of business processes defined in BPEL4WS onto OWL-S based process ontology, which is a container framework to describe Web services in form of workflows [13]. The tool supports multiple WSDL files which are along with a BPEL4WS file. WSDL files are distinguished in terms of slave WSDL and master WSDL. The master WSDL

is the main one that all slave WSDL(s) refer to. WSDL serves as the foundation of this mapping process. It describes all the data that used in a business process. BPEL4WS explains the business process specifically and accurately. Accordingly we build our process structure in OWL-S closely correspondent to BPEL4WS but in a totally different way. BPEL4WS, as a typical workflow language, uses flow to describe the business process while OWL-S as an ontology language converts the process to a hierarchical structure which is similar to the object-oriented programming concept.

WSDL messages are used to represent the abstract definition of the data being transmitted in and out of the processes [4]. WSDL messages consists one or more logical parts. Each part is associated with a type from some type system (data type defined or built-in XML Schema: XSD). Since we know the data types of parts of the messages by the *type* attribute of the part element, all the messages will be represented as OWL class and the type of the class will not be restricted to any particular data type but will use OWL object (i.e. *Thing*) as the data type. Parts of each message will be mapped onto the properties of their corresponding OWL class and the data type of the property will be based on the type specified for that part. Since WSDL messages use the XSD type system for associating the data types [4] with its parts, we can have customised and built data types defined in the XML schema declaration used by WSDL messages. To use these 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. This will be done by using the namespaces declared for the schema declarations in the input WSDL file for its internal use and then importing them in our output process OWL file.

Variables in BPEL4WS provide the means for holding messages that constitute the state of a business process. The messages held are often those that have been received from the partners or are to be sent to the partners via primitive activities. The type of each variable may be a WSDL message type, 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 WSDL message thus sharing the same data type that of the WSDL message it refers to. Hence, such types of variables are mapped onto the data types in the OWL-S process ontology (same as WSDL messages mapped). Variables defined by the *messageType* attribute (representing a WSDL message type) in a BPEL4WS process will be mapped in the same way as the WSDL messages mapped onto the data types in OWL-S process ontology using the OWL class. We will avoid duplicate declarations by mapping the variables to OWL class and then making the class same as the WSDL message (data type already declared) it represents; using the *sameClassAs* element of OWL. The variables defined using the *type* or *element* attributes will also be

¹ <http://www.it.swin.edu.au/centres/cicec/bpel2owls.htm>

mapped in the same way (using OWL class) but will not restrict themselves to any particular data type (XML schema simple type or element). The data types of such variables will be the OWL object (i.e. *Thing*) by default.

The inputs and outputs of the atomic processes derived from <receive>, <reply> and <invoke> activities accordingly will be described in the Data Flow OWL using 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 (represents that two parameters used for referencing are equal), used similarly as above for referencing the inputs and outputs of the derived atomic processes from primitive activities to their corresponding super class atomic process's inputs and outputs. Furthermore, we represent the referencing of the inputs and outputs of the derived composite processes from structured activities in the Data Flow OWL using the OWL-S *valueOf* class thus, representing the complete data flow for both atomic processes and composite processes derived from BPEL4WS primitive and structured activities respectively.

From the data-flow point of view the relationships between the atomic classes that beneath the process can be implicitly indicated. After the description of the composite process, the atomic processes are described one by one.

If we re-visit the example mentioned in Section 3, in WSDL/BPEL environment, input and output messages will be defined in related WSDL files like:

```
<portType name="AppExamPT">
  <operation name="check">
    <input message="def:AnswerSheet"/>
    <output message="def:Score"/>
  </operation>
</portType>
```

With OWL-S service profile support, more explicit syntax and semantics will be incorporated:

```
<process:AtomicProcess rdf:ID="check">
  <process:hasInput>
    <process:Input rdf:ID="AnswerSheet1">
      <process:parameterType
        rdf:resource="#AnswerSheet"/>
    </process:Input>
  </process:hasInput>
  <process:hasOutput>
    <process:UnConditionalOutput
      rdf:ID="Score1">
      <process:parameterType
        rdf:resource="#Score"/>
    </process:UnConditionalOutput>
  </process:hasOutput>
</process:AtomicProcess>
```

5. Related Work

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 [2, 15]. There is a growing trend that the next generation of workflow systems will be built in a truly decentralised manner. Serendipity-II [6] 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 for distributed software development projects. More specifically, the peer-to-peer (p2p) based workflow systems have been recognised as one of the most strategic future directions for workflow research [10]. An ongoing p2p-based workflow project is conducted at Manchester Metropolitan University. This project 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) [5]. 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 [15,17].

Research groups at Stanford University and Carnegie Mellon University have led the work in adapting BPEL4WS or WSDL for semantic Web or OWL-S [8, 11]. Till now, such work only maps WSDL service descriptions to 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 [11], only mapping of port types and operations to their corresponding DAML-S atomics processes is presented. It does not reflect the mapping of WSDL messages in DMAL-S process ontology. We extended the mapping of WSDL2DAML-S, by including the WSDL messages in our OWL-S based workflow process ontology. Our work reported in [13] is a significant complementary work to the above mentioned efforts. It is also a good stand-by support tool for SwinDeW-S in service profile creation by integrating BPEL4WS, XPDL, WSDL with OWL-S together.

6. Conclusions and Future Work

In this paper we discussed service profile issues in the context of SwinDeW-S, a Web service 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 have shown their incompetency and inflexibility if deployed along with traditional extended XPDL-based service profile

descriptions. Thus a new ontology language for Web services, OWL-S, has been deployed. With OWL-S, especially the tools that map BPEL4WS/WSDL to 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 feasible in the future.

Acknowledgement

The research work reported in this paper is partly supported by Swinburne Vice Chancellor's Strategic Research Initiative Grant 2002-2004. The IBL student, Jonathan Derham, has contributed significantly to the implementation of SwinDeW-S.

Reference

- [1] 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.
- [2] 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.
- [3] 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/Webservices/library/ws-bpel>, 2003.
- [4] 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.
- [5] G. J. Fakas and B. Karakostas, A Peer to Peer Architecture for Dynamic Workflow Management, *Information and Software Technology Journal*, Elsevier, 46(6): 423-431, 2004.
- [6] 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.
- [7] S. Liu, A. Goh and E. Soong, A Formal Framework to Support Workflow Adaptation, *Int. J. of Software Engineering and Knowledge Engineering* 12(3): 245-268, June, 2002.
- [8] 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 (ISWC2003)*, LNCS 2870, 227-241, 2003.
- [9] D. Martin et al (eds) *OWL-S 1.1 Release, Upper ontology for Services*, <http://www.daml.org/services/owl-s/1.1/>, Nov. 2004.
- [10] 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.
- [11] 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
- [12] J. Shen and Y. Yang, Extending RDF in Sistributed Knowledge-Intensive Applications, *Future Generation Computer Systems*, 20(1): 27-46, 2004.
- [13] J. Shen, Y. Yang, B. Lalwani. Mapping Web Services Specifications to Process Ontology: Opportunities and Limitations, *Proc. of 10th IEEE International Workshop on Future Trends in Distributed Computing Systems (FTDCS'04)*, 229-235, 2004.
- [14] 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.
- [15] J. Yan, Y. Yang and G. K. Raikundalia, A Decentralised Architecture for Workflow Support, *Proc. of the 7th International Symposium on Future Software Technology (ISFST'02)*, CD ISBN: 4-916227-14-X, Wuhan, China, Oct. 2002.
- [16] J. Yan, Y. Yang and G. K. Raikundalia, A Data Storage Mechanism for P2p-based Decentralised Workflow systems, *Proc. of 15th Int. Conf. on Software Engineering and Knowledge Engineering (SEKE2003)*, 354-358, SF, USA, July 2003.
- [17] J. Yan, Y. Yang and G. K. Raikundalia, Enacting Business Processes in a Decentralised Environment with P2p-based Workflow Support, *Proc. of 4th Int. Conf. on Web-Age Information Management (WAIM2003)*, LNCS 2762, 290-297, Chengdu, China, Aug. 2003.
- [18] J. Yan, Y. Yang and G. K. Raikundalia, Decentralised Coordination for Software Process Enactment, *Proc. of 9th European Workshop on Software Process Technology (EWSP03)*, LNCS 2786, 164-172, Helsinki, Finland, Sept. 2003.
- [19] J. Yan, Y. Yang and G. K. Raikundalia, Critical Issues in Extending P2p-based SwinDeW System for Incomplete Process Support, *Proc. of 8th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2004)*, 312-317, Xiamen, China, May 2004.