A Multi-agent Based Workflow System Embedded with Web Services

Maryam Purvis, Bastin Tony Roy Savarimuthu, Martin Purvis
Department of Information Science, University of Otago, PO Box 56, Dunedin, New Zealand
{tehrany, tonyr, mpurvis}@infoscience.otago.ac.nz

Abstract

The rapid changes in business environment call for flexible and adaptive workflow systems. In this paper we describe how we incorporate Web Services as well as provide an intelligent resource allocation mechanism to our existing agent-based workflow management system (WfMS). The extended architecture enhances the adaptability aspect of our system. In addition to the process perspective of adaptability it also provides support for resource and task perspectives. We discuss how Web Services can be located and the services provided by them can be invoked. We also discuss how the agents and Web Services can co-exist in order to create highly flexible and dynamic WfMS.

Using an example we show how our framework can be used to achieve adaptability from a process, resource and task perspectives using both agents and Web Services.

1. Introduction

Workflow management systems (WfMS) [5,15,19] are increasingly being used to manage business processes associated with distributed global enterprises. Some of the benefits of using a WfMS are

- Ability to visualize the overall process and interdependencies between various tasks.
- Automation of the processes.
- Coordination and collaboration between various business entities.

Existing, commercially available, workflow management systems (WMS) do not offer sufficient flexibility for distributed organizations that will be participating in the global market. These systems have rigid, centralised architectures that do not operate across multiple platforms [20]. Improvements have been made by employing a distributed network of software agents that can adapt to changing circumstances [28, 29].

In the past, WfMSs were used in well-defined activities, such as manufacturing, where the processes tend to be more established and stable. But in the current climate WfMS may be used in connection with more fluid business processes, such as e-commerce, or in processes involving human interactions, such as the software development process. In such situations, at times, it is not always possible to predict in advance all the parameters that may be important for the overall processes. In particular some of the reasons for requiring an adaptive WfMS are as follows [3,4,7]:

- It may not be possible to specify all the process details associated with a complex process at the outset. The initial model may represent a high level view of the process, which includes some of the sub processes. Gradually some of these sub processes may be refined as the stakeholders obtain more experience and knowledge of a particular process.
- Due to changes in the market, a new requirement may be imposed which can impact the process definition.
- The workflow could use services provided by the external entities such as the Web Service provided by banks, stock markets etc.

The workflow system is adaptable in three ways: 1) From the process perspective in which the process model can be changed dynamically and 2) from the resource perspective in which the choosing a particular resource could be done at run time based on the history data and from 3) individual task/activity point of view where an appropriate Web Service can be invoked on the fly.

Adaptability from a process perspective has been discussed in our previous work [28,29]. In this paper we discuss how the workflow is adaptable from the resource perspective and the individual activity point of view, which involves the incorporation of intelligent mechanism in selecting a suitable resource dynamically and the integration of Web Services to the workflow.

2. Background

2.1. Coloured Petri Nets

Coloured Petri nets are used to model workflow systems, due in part to their sound mathematical foundation and to the fact that they have been used extensively for modelling distributed systems [13].

Coloured Petri nets consist of the following basic elements:

- *Tokens,* which are typed markers with values - the type can in our implementation be any Java class.
- *Places* (circles), which are typed locations that can contain zero or more tokens.
- *Transitions* (squares), which represent actions whose occurrence (firing) can change the number, locations and value of tokens in one or more of the places connected to them. Tokens may have guards, which must evaluate to TRUE in order for a transition to fire. In a workflow model a transition may represent a task.
- *Arcs* (arrows) connecting places and transitions. An arc can have associated inscriptions, which in our implementation are Java expressions whose evaluation to token values affects the enabling and firing of transitions.

Some reasons for preferring Petri net modelling to other notations in connection with workflow modelling are [2]:

- They have *formal semantics*, which make the execution and simulation of Petri net models unambiguous. It can be shown that Petri nets can be used to model workflow primitives identified by the Workflow Managment Coalition (WfMC) [1]
- Unlike some event-based process modelling notations, such as dataflow diagrams, *Petri nets can model both states and events*.
- There are *many analysis techniques* associated with Petri nets, which make it possible to identify 'dangling' tasks, deadlocks, and safety issues.

Currently, we are using JFern [17] - a CPN-simulator and enactment engine to design and execute the models. We have modified the JFern editor to support hierarchical models, which enables the modeler to start with a coarse model of the process and gradually refine each of the subprocesses into a separate Petri net model.

2.2. Agent systems

2.2.1. WfMS using software-agents

Agent systems provide an open, flexible and distributed framework [8,21,23,25]. In the context of WfMS, agent technology has been used in different ways [14,20]. In some cases the agents fulfil particular roles that are required by different tasks in the workflow. In these cases the existing workflow is used to structure the coordination of these agents [12,16]. An example of this approach is the work by M. Nissen in designing a set of agents to perform activities associated with the supply chain process in the area of E-Commerce [16].

In other cases, the agents have been used as part of the infrastructure associated with the WfMS itself in order to create an agent-enhanced WfMS [22, 24]. We have used our agent technology for this latter case. These agents provide an open system with loosely coupled components, which provides more flexibility than the traditional systems. As part of the agent framework, each agent can be located in a separate host over the Internet, which enables the distribution of the workflow-related activity and reduces the problems associated with overloading a particular system component.

Some researches have combined both of these approaches [9], where an agent-based WfMS is used in conjunction with specialized agents that provide appropriate application-related services.

Our research is focussed on developing an agent-enhanced WfMS, where the work associated with running a WfMS has been partitioned among various agents that are interacting with each other by following standard agent communication protocols. For example, the model associated with a business process that is represented with the Coloured Petri net formalism is executed by a specially designed agent for this purpose. This framework provides an environment where new process models (when they are required) can be dynamically incorporated into the system and also the process model itself can be distributed over the net. The distribution of the process model is important in situations where the processes associated with a large organization with various departments (which might be located in different sites) are modelled, and inter-process interactions are required. In these cases each site might be running its local process model that is one component of the overall process model associated with the organization.

2.2.2. Opal

Opal [18] is a java-based agent platform developed at the University of Otago since 2000. It meets the standards of the Foundation for Intelligent Physical Agents (FIPA) [11] for agent platforms and incorporates a modular approach to agent development.

In Opal every FIPA ACL-speaking [10] agent consists of several micro agents. This facilitates the development of agents, because one can easily add or remove micro agents to an Opal agent and thereby change the behaviour of this “macro”-agent.

2.3. Web Services

Web Services are software components available on the internet, that provide certain services which may be of

general interest such as weather monitoring services, currency converters etc. These services may be of interest for some of the tasks associated with a process model. For example, the process model associated with the travel plan of the tourist depends upon the environmental factors such as the weather conditions. The task associated with finding the weather condition can be provided using a publicly available Web Service.

2.4. Former approaches for adaptive WfMS

The need for adaptive workflow systems and possible solutions to these problems have been discussed by researchers [2, 4, 27, 34]. A few researchers have provided solutions to some aspects of adaptability [2, 27]. But, only a few researchers have tackled the problem associated with the running work cases [2]. Our previous papers [33, 28, 29] addressed this issue for some scenarios.

Some researchers have integrated agent based workflow systems with Web Services [26, 27, 31,32]. However some enhancements can be made to improve these approaches.

1. In the research done by Paul et.al [31,32], BPEL4WS has been used as a process model and this model is converted to Petri net. The problem in this approach as acknowledged by the authors is that, BPEL4WS does not support selective routing.

However, in our system the process model is described using coloured Petri net that can be directly executed.

- Firstly, we do not have the selecting routing problem.
- Secondly, the conversion mechanism of a BPEL4WS model to a Petri net model has to be validated to ensure the structural and behavioural equivalence associated with the original model.

2. Our system provides a dynamic incorporation of the process models, which does not seem to be possible in their system. Their system provides adaptability for a particular task in the process but not the process model as a whole.

3. Enhanced architecture of the system

3.1. The architecture

Our system is based on Opal [18] and uses the CPN-execution tool JFern [17] Our system consists of seven special opal agents which provide the functionality to control the workflow. Figure 1 contains these agents and their relationship to each other.

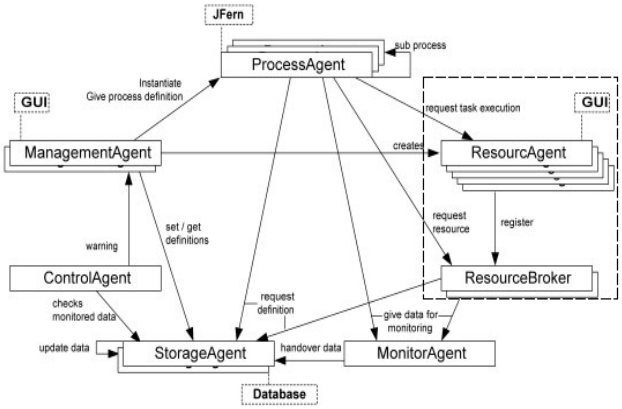

Figure 1: Architecture of the agent based workflow system

The manager agent provides all functionality the workflow manager needs such as creation and deletion of tasks, roles and process definitions, instantiation of new process instances and creation of resource agents. The process agent executes a process instance. Each resource in the system has its own resource agent. Every resource in the system gets registered to one of the broker agents that allocate the resources to the process. The storage agent manages the persistent data that is needed. The monitor agent collects all the process specific data and sends them to the storage agent. The control agent continuously looks for anomalies to the criteria specified by the human manager and reports the violations to these criteria to the manager agent. The manager agent provides information to the human manager, which can be used for the feedback mechanism.

3.2. Enhancements to the framework

The enhancements to the framework are two fold. a) The resources allocated for a particular task is done based on the history data and b) dynamic discovery of Web Services for a particular task and connecting to the Web Service through agents. The dashed region on the above diagram (Figure 1) indicates the components of the system that required to be modified. The functionalities of the resource and resource broker agents had to be modified to accommodate the above changes.

3.2.1. Intelligent resource allocation

For each case that is being enacted the data pertaining to the tasks such as the task execution time, waiting time to obtain a resource, time taken by the resource to execute the task etc. are stored persistently in distributed databases using the monitoring agent [30]. Figure 2 shows overall utilization of the resources and Figure 3 shows the waiting time of various tasks for similar cases. These are the kind of information that have been stored persistently and

which is of interest during the process of resource allocation.

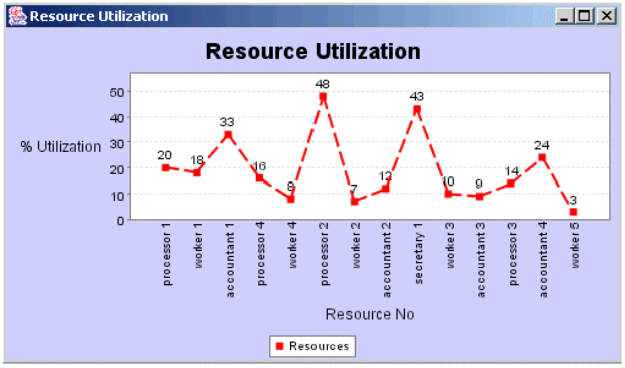

Figure 2: Percentage utilization of resources (resource instances in x axis and % utilization in y axis)

Our architecture supports the allocation of resources to tasks depending upon these history data. By examining the data that has been collected, the resource that performs a task the best can be obtained. If this resource is available at that point of time, it is assigned the responsibility of completing the task. Otherwise the second best available resource is selected.

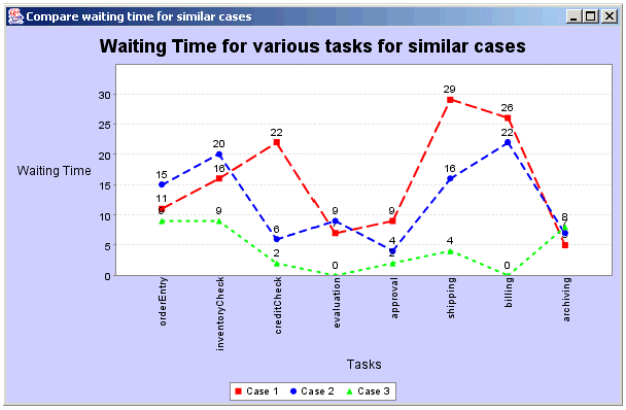

Figure 3: Waiting time for various tasks for 3 similar cases (Tasks in x axis and waiting time in y axis)

3.2.2. Dynamic discovery and invocation of Web Services

The resources in our workflow system are of three types or roles. The first kind of resources can perform tasks such as 'billing an invoice'. The second kind of resource can connect to a device such as a printer, scanner or a PDA. The third kind of resource can connect to a Web Service and fetch the results.

Our framework supports the dynamic discovery of Web Services by connecting to a UDDI service and invoking an appropriate service. This is achieved by the combination of SOAP, WSDL and UDDI.

3.3 Adaptability

Our framework supports adaptability form process, resource and task perspectives.

3.3.1 Process Perspective

As was described in the earlier work [29], for each work case a new process agent is created and an appropriate CPN model is instantiated. The work case is represented by a token in the CPN model. While the workflow system is running there might be a scenario in which requires a new CPN model (due to change in the business process). In this case there are several possible actions that can take place. The choice of these actions depends on the scope of the change requested and the extent to which it has to be applied to the existing work cases. In case the change has to be applied to new work cases waiting in the queue to be processed, then we can easily instantiate the process agent with the new or modified model instead of the old model for these work cases. However, if the proposed change in the process should be applied to the running instances, it is necessary to make sure that the change does not violate the structural and semantic consistency of the model before we can transfer the state of the running instance to the new model. To decide whether a running case can be replaced using a new model, we use an algorithm [33] inspired by van der Aalst. More details associated with this process are described in our previous works [28,29].

3.3.2 Resource perspective

The resource adaptability is achieved by the run time binding of the resource agents to the resource broker agent and allocation of a particular resource agent to perform a task dynamically. At any point of time, a resource may register its availability and the roles it can perform.

3.3.3 Task perspective

The tasks specified in the model, when necessary can invoke a suitable Web Service. The behaviour of these services can change over time without requiring any change to be made to the module that references them.

In the process perspective, the whole model can be replaced by a new model. Task perspective deals with the atomic activities specified in the process model such as 'billing' task associated with a book-purchasing model.

The adaptability of our system provides a hybrid architecture which allows the integration of the legacy systems coupled with Web Services and multi agents.

4. Example

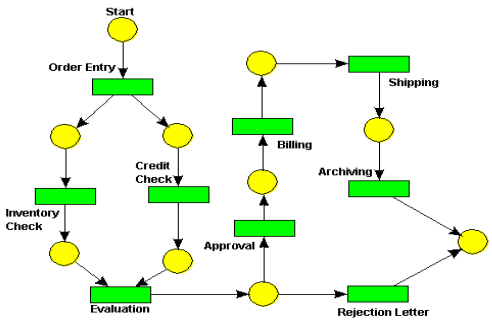

Figure 4: The adapted process definition

4.1. Simple example using intelligent resource allocation

To illustrate the functionality of the system, we choose a simple process of ordering a book. The CPN-model of this is shown in Figure 4. After an order arrives, the company has to check if a copy of this book is in the inventory and whether the credit rating of the customer is satisfactory. This credit checking activity is available as an external Web Service, which can be invoked by a resource agent. If the credit of the customer is below the purchasing limits, the order is rejected. Otherwise the execution continues to the next set of tasks. Credit and inventory checks are done in parallel to speed the process up and the results are then evaluated. Assuming that the processing of the request has been approved, the shipping of the book and the sending of the bill are done in parallel. Finally the results of shipping and billing activities are archived to be able to follow up possible complaints.

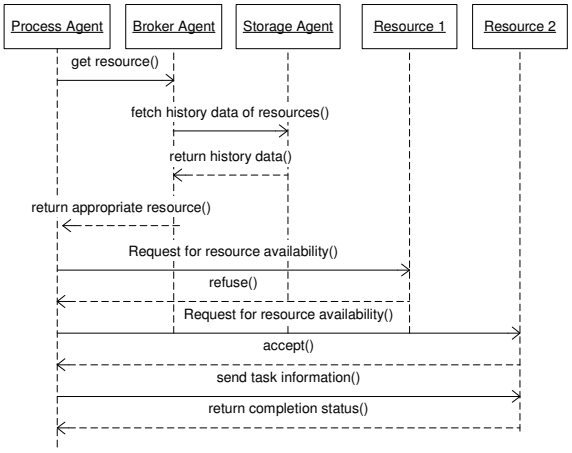

Figure 5 showing the allocation of resource to a task intelligently.

Assume that a new order arrives. The first step is the creation of a new process agent and getting all task definitions. Then the process agent *process1* starts executing the work case by putting a job-token in the place called *Start* (Figure 4). This activates the transition *Order Entry*. According to the task definition, a resource of the role *processor* is needed. The process agent has therefore to ask the resource broker for a resource of the role *processor*.

The process agent requests the broker agent to return a resource that can perform this task. The broker agent first finds the list of all resources, which can perform the task from the storage agent. It also finds the resources that are idle at that time out of this list. Then, depending upon the history data for that task, it chooses the resource, which has performed that task in the least amount of time. Figure 5 shows how the resource is allocated depending upon the history data.

If it cannot allocate a resource it will send a message to the process agent that it failed to allocate a resource and the process agent has to decide how to handle this (for instance ask another resource broker, or wait a certain time and ask the same broker again).

After the resource executes the task and returns the result of the task, the Jfern engine gets the information that the task *Order Entry* has been successfully executed and continues executing the CPN-model. This process continues for the remaining tasks based on the model.

4.2. Demonstration of dynamic discovery and invocation of Web Services

There are scenarios in a workflow model where certain services are available in the Internet, which can be utilized. In our example, the credit of the customer can be checked using a Web Service. So, the workflow system has to enable mechanisms by which this can be accomplished. Our framework supports the discovery of such Web Services and their invocation.

4.2.1 Dynamic discovery of Web Services

Figure 6 shows how the services are discovered dynamically by our workflow system. The process agent contacts the resource broker agent to allocate appropriate resource. Depending upon the 'role type' requested by the process agent, appropriate agents are instantiated. In this case an agent capable of connecting to a Web Service is instantiated. Also, the broker agent connects to the externally available UDDI services using UDDI4J and gets a list of services. In our example, to perform the 'credit check' task, it gets the Web Services provided by a published service. In case of multiple services are available, the system displays all available services and the human manager can select an appropriate service. The broker agent returns the instantiated resource agent and

the URI that needs to be invoked by the newly created resource agent so that the Web Service can be accessed.

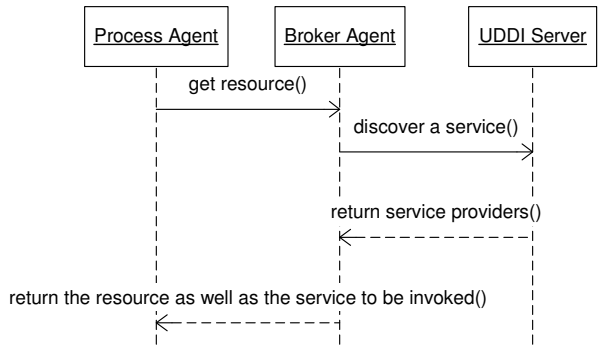

Figure 6 Dynamic discovery of Web Services

4.2.2 Invocation of Web Services

The process agent sends the URI to the resource agent and the resource agent contacts the Web Service by providing appropriate parameters as specified by the WSDL definition of the service interface. An example of a WSDL for a credit checking web service is given below.

```
...
<message name="creditRequest">
  <part name="customerID" type="xsd:string"/>
</message>
<message name="creditResponse">
  <part name="creditResult" type="xsd:string"/>
</message>
 :
<portType name="CheckCredibility">
  <operation name="checkCredit">
    <input message="tns:creditRequest"/>
    <output message="tns:creditResponse"/>
  </operation>
</portType>
...
```

Figure 7: A sample WSDL code segment for credit checking service

The WSDL document shown in Figure 7 describes an operation called *checkCredit.* In order to obtain the credit of a particular customer the variable named *customerID* is assigned with a string value and this information is sent as a *creditRequest* message. After the request is submitted to the service provider, it processes the request and returns the result to the variable called *creditResult* as specified in the *creditResponse* message.

The resource agent parses the corresponding WSDL document using WSDL4J and invokes the Web Service using appropriate arguments as described above. The agents use Apache Axis, to invoke the Web Services.

After the Web Service returns the result, the resource agent returns the result to the process agent. The process agent then interprets the result and decides where the token in the Petri net has to be placed, i.e., if the credit is low, then the work case is rejected or else it proceeds to the execution of the next task by enabling the *Approval* transition. Figure 8 shows the invocation of Web Services.

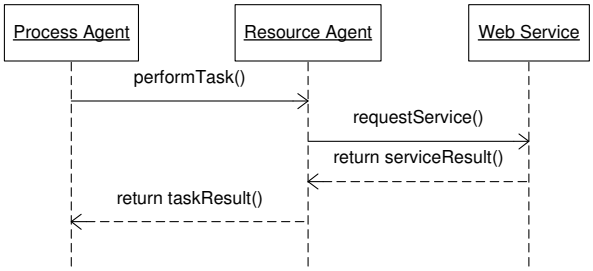

Figure 8: Invocation of Web Services

5. Conclusion

In this paper we have described the enhanced architecture of a flexible system, which can be distributed and can deal with various levels of adaptability from a process perspective, resource perspective and task perspective. We have shown how our agent-based system allocates resources based upon the past data and also how it can discover and connect to the Web Services dynamically.

In order to identify and use the externally available resources more meaningfully, we need to investigate further on the integration of semantic web in our system.

We also intend to integrate into the currently implemented framework one of the existing CPN-model analysis tools so that we can examine the model for certain properties such as soundness and reachability.

6. Acknowledgments

The authors wish to acknowledge the work of Lars Ehrler and Martin Fleurke in the implementation of the original system. The authors wish to thank Marius Nowostawski for his work in implementing the OPAL and JFern systems.

7. References

1. *The Workflow Reference Model, Document No. TC00-1003, Issue 1.1.*. 1995, Workflow Management Coalation.
2. Aalst, W.M.P.v.d. *Three Good Reasons for Using a Petri-net-based Workflow Management System.* in *International Conference on Practical Aspects of Knowledge Management (PAKM'96), Workshop on Adaptive Workflow*. 1996. Basel, Switzerland.
3. Aalst, W.M.P.v.d., *Exterminating the Dynamic Change Bug: A Concrete Approach to Support Workflow Change.* Information Systems Frontiers, 2001. **3**(3): p. 297-317.
4. Aalst, W.M.P.v.d., Basten, T., Verbeek, H.M.W., Verkoulen, P.A.C., and Voorhoeve., M. *Adaptive Workflow: An Approach Based on Inheritance.* in *Proceedings of the IJCAI'99 Workshop on Intelligent Workflow and Process Management: The New Frontier for AI in Business*. 1999. Stockholm, Sweden.
5. Aalst, W.M.P.v.d. and Hee, K.v., *Workflow Management: Models, Methods, and Systems.* 2002, Cambridge, MA, USA: MIT Press.
6. Aalst, W.M.P.v.d., Hofstede, A.H.M.t., Kiepuszewski, B., and Barros., A.P., *Workflow Patterns*. 2002, Queensland University of Technology: Brisbane, Australia.
7. Bandinelli, S., Fuggetta, A., and Ghezzi, C., *Process Model Evolution in the SPADE Environment, Technical Report No. 14 , ESPRIT-III Project GOODSTEP (6115).* IEEE Transactions on Software Engineering, 1993. **19**(12): p. 1128-1144.
8. Bradshaw, J., *An Introduction to Software Agents*, in *Software Agents*, J. Bradshaw, Editor. 1997, MIT Press: Cambridge. p. 3-46.
9. Chen, Q., Hsu, M., Dayal, U., and Griss:, M.L. *Multi-agent cooperation, dynamic workflow and XML for e-commerce automation*. in *the proceedings of the fourth international conference on Autonomous agents*. 2000. Barcelona, Spain: ACM Press.
10. FIPA, *FIPA Communicative Act Library - Specification*. 2002. http://www.fipa.org/specs/fipa00037/
11. FIPA, *http://www.fipa.org*.
12. Jennings, N.R., Faratin, P., Norman, T.J., O'Brien, P., and Odgers, B., *Autonomous Agents for Business Process Management.* Int. Journal of Applied Artificial Intelligence, 2000. **14**(2): p. 145-189.
13. Jensen, K., *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use, Vol. 1: Basic Concepts.* EATCS Monographs on Theoretical Computer Science. 1992, Heidelberg, Berlin: Springer-Verlag GmbH. 1-234.
14. Joeris, G., *Decentralized and Flexible Workflow Enactment Based on Task Coordination Agents*, in *Proc. of the 2nd Int'l. Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS 2000 @ CAiSE*00)*. 2000: Stockholm, Sweden. p. 41-62.
15. Meilin, S., Guangxin, Y., Yong, X., and Shangguang, W. *Workflow Management Systems: A Survey*. in *Proceedings of IEEE International Conference on Communication Technology*. 1998. Beijing, China.
16. Nissen, M.E. *Supply Chain Process and Agent Design for E-Commerce*. in *33rd Hawaii International Conference on System Sciences*. 2000. Maui, HI, USA.
17. Nowostawski, M., *JFern - Java-based Petri Net framework*. 2003.
18. Purvis, M.K., Cranefield, S.J.S., Nowostawski, M., and Carter, D., *Opal: A Multi-Level Infrastructure for Agent-Oriented Software Development*. 2002, Department of Information Science, University of Otago: Dunedin, New Zealand.
19. Schael, T., *Workflow Management Systems for Process Organisations*. Lecture Notes in Computer Science. Vol. LNCS 1096. 1998, Berlin, Germany: Springer-Verlag.
20. Shepherdson, J.W., Thompson, S.G., and Odgers, B. *Cross Organisational Workflow Co-ordinated by Software Agents*. in *CEUR Workshop Proceedings No 17. Cross-Organisational Workflow Management and Co-ordination*. 1999. San Francisco, USA: CEUR.
21. Shoham, Y., *An Overview of Agent-Oriented Programming*, in *Software Agents*, J. Bradshaw, Editor. 1997, MIT Press: Cambridge. p. 271-290.
22. Stormer, H. *AWA - A flexible Agent-Workflow System*. in *Workshop on Agent-Based Approaches to B2B at the Fifth International Conference on Autonomous Agents (AGENTS 2001)*. 2001. Montreal, Canada.
23. Sycara, K.P., *Multiagent Systems.* AI magazine, 1998.
24. Wang, M. and Wang, H. *Intelligent Agent Supported Flexible Workflow Monitoring System*. in *Advanced Information Systems Engineering: 14th International Conference, CAiSE 2002*. 2002. Toronto, Canada: Springer Verlag GmbH.

25. Wooldridge, M.J., *Intelligent Agents*, in *Multiagent Systems*, G. Weiss, Editor. 1999, MIT Press: Cambridge. p. 27-77.
26. José M. Vidal, Paul Buhler, and Christian Stahl. Multiagent systems with workflows. *IEEE Internet Computing*, 8(1): 76-82, January/February 2004.
27. Paul Buhler, José M. Vidal, and Harko Verhagen. Adaptive workflow = Web Services + agents. In *Proceedings of the International Conference on Web Services*, pages 131-137. CSREA Press, 2003.
28. Fleurke, M and Ehrler, L, Purvis, M. A. (2003). "*JBees - An Adaptive and Distributed Agent-based Workflow System*", in Proceedings of the International Workshop on Collaboration Agents: Autonomous Agents for Collaborative Environments (COLA 2003), Halifax, Canada, October 2003. IEEE/WIC Press. Ghorbani, A. And Marsh, S., Ed.
29. Ehrler, L., Fleurke, M., Purvis, M. A**.** and Savarimuthu, B.T.R., "Agent-Based Workflow Management Systems(Wfmss) : JBees- A Distributed and Adaptive WFMS with Monitoring and Controlling Capabilities", to be published in a special issue of the *Journal of Information Systems and e-Business on Agent-Based Information* (2005).
30. Savarimuthu, B.T.R., Purvis, M. A. and Fleurke, M. (2004), "Monitoring and Controlling of a Multi-agent Based Workflow System", *Proceedings of the Australasian Workshop on Data Mining and Web Intelligence (DMWI2004)*, Conferences in Research and Practice in Information Technology, Vol. 32, Australian Computer Society, Bedford Park, Australia (2004) 127-132.
31. Paul Buhler and José M. Vidal. Integrating agent services into BPEL4WS defined workflows. In *Proceedings of the Fourth International Workshop on Web-Oriented Software Technologies*, 2004.
32. Paul Buhler and José M. Vidal. Enacting BPEL4WS specified workflows with multiagent systems. In *Proceedings of the Workshop on Web Services and Agent-Based Engineering*, 2004.
33. Martin Fleurke. JBees, an adaptive workflow management system - an approach based on petri nets and agents. Master's thesis, Department of Computer Science, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands, 2004.
34. Dr. N.C. Narenda, Adaptive Workflow Management . An integrated Approach and System Architecture, Proceedings of the 2000 ACM symposium on Applied computing, March 2000.