Evaluation of a multi-agent based workflow management system modeled using Coloured Petri Nets

Maryam Purvis, Bastin Tony Roy Savarimuthu and Martin Purvis

Department of Information Science, University of Otago,
P O Box 56, Dunedin, New Zealand
{tehrany,tonyr,mpurvis}@infoscience.otago.ac.nz

Abstract. Workflow management systems (WfMS) should address the needs of rapidly changing business environments We have built a multi-agent based framework, JBees, which addresses these needs. We evaluate our agent based workflow system modelled using Petri nets with the proposed standards for various workflow patterns and communication patterns. The coloured Petri net model supports the workflow patterns and the FIPA compliant agent framework supports the communication patterns. The agent based communication patterns along with the workflow patterns make the workflow management system equipped with a comprehensive set of capabilities such as adaptability and distribution.

1 Introduction

Most of the commercially available workflow management systems do not offer sufficient flexibility for distributed organizations that participate in the global market. These systems have rigid, centralized architectures that do not operate across multiple platforms [8]. Employing a distributed network of autonomous software agents that can adapt to changing circumstances would result in an improved workflow management system. 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 for more fluid business processes, such as e-commerce, or in processes involving human interactions, such as the software development process. In such situations, it is not always possible to predict in advance all the parameters that may be important for the overall processes. This gives rise to the need of adaptive systems. Our previous works ([2] and [17]) describe the issues addressed by our agent-based framework JBees.

In this paper we evaluate our agent based workflow system in two ways. We first evaluate our system for the various workflow patterns. Secondly we compare the communication patterns supported by our system. These comparisons are with reference to the patterns described in the previous work by van der Aalst ([10],[11],[12],[13],[14] and [15]). The paper is organized as follows. A brief description of various notations used and our agent-based framework are given in section two. The third section describes how we evaluate our system using various workflow and communication patterns. The concluding remarks are presented in the fourth section.

2 Background

In this section we explain the background of our work, which includes the coloured Petri nets, which are used to design the process models and the multi-agent system on which our workflow system has been built.

2.1 Coloured Petri Nets

The sound mathematical foundation behind the Coloured Petri nets (CPN) [16] makes it a very useful tool for modelling distributed systems. Petri nets consist of four basic elements. The *tokens* which are typed markers with values, the *places* that are typed locations that can contain zero or more tokens, the *transitions* which represent actions whose occurrence can change the number and value of tokens based on the expression specified on the arcs in one or more of the places connected to them and the *arcs* that connect places and transitions. Some reasons for preferring Petri net modeling to other notations used for workflow modeling are given by [20]:

– 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 Management Coalition (WfMC) [21]
– Unlike some event-based process modeling notations, such as dataflow diagrams, Petrinets can model both states and events.
– There are many analysis techniques associated with Petrinets, which make it possible to identify 'dangling' tasks, deadlocks, and safety issues.

2.2 Agent Systems

According to Sycara [25], there are several benefits of using multiagent systems for building complex software. For example, multiagent systems can offer a high level of encapsulation and abstraction. Some commonly accepted characteristics of agents are listed in [22,23, 24]. Because agents are independent, every agent can decide by itself what is the best strategy for solving its particular problem. The agents can be built by different developers; as long as they understand the communication, they can work together. A second important benefit is that multiagent systems offer a distributed and open platform architecture. Agents can support a dynamically changing system without the necessity of knowing each part in advance. This requires, however a matchmaking infrastructure. Our system is based on the Java-based agent platform Opal[7], developed at the University of Otago since 2000. It meets the standards of the Foundation for Intelligent Physical Agents (FIPA) [6] for agent platforms and incorporates a modular approach to agent development [26].

2.3 Related Work

In the context of WfMSs, agent technology has been used in different ways [27]. In some cases the agents fulfill 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[28,29]. 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 [29].

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 [30,31]. These agents provide an open system with loosely coupled components, which provides more flexibility than the traditional systems. Some researchers have combined both of these approaches [32], where an agent-based WfMS is used in conjunction with specialized agents that provide appropriate application-related services. We have taken the latter approach which provides sufficient flexibility required for a dynamic and adaptive system.

Adaptive workflows have been discussed for many years and many people have described what should be done[10, 32]. Only a few have proposed techniques to manage adaptability and only a small number of actual implementations have been made that tackle some aspects of adaptability [32]. Transferring running work cases to a new model is still a difficult issue. The work done in the paper [32] describes a prototype, which provides some adaptability by manual transfer of tokens in the new process model. This is indicated in a comparison of current WfMS that was done by Van der Aalst et al.[13].

2.4 Architectural overview

Our research is focused on developing an agent-enhanced WfMS, where the work associated with running a WfMS has been partitioned among various collaborating agents that are interacting with each other by following standard agent communication protocols. JBees is based on Opal [7] and uses the CPN execution tool JFern [5]. A first description of JBees can be found in the previously published papers [2] and [17]. Our enhanced system consists of seven Opal agents, which provide the functionality to control the workflow. Figure 1 shows these seven agents and their collaboration.

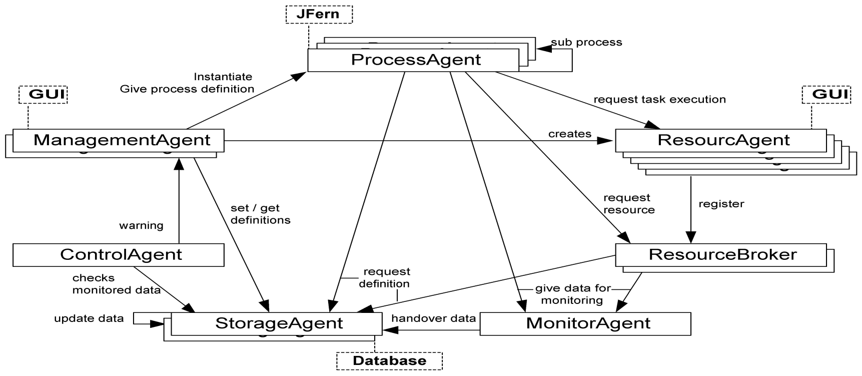

Figure 1: Showing the architecture of JBees

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.

2.5 Flexibilities of our workflow system

The flexibilities of our workflow system enable us to provide the support for distribution, adaptability, monitoring and controlling of processes. JBees supports the inter-organizational co-operation through the distribution of process. For example, the main process could be present in Germany and the sub process could be present in New Zealand. The use of multi-agent facilitates the distribution of processes. Also, the persistent data can also be distributed. The user of the system can decide to modify or change a running process [1]. Our system has also been endowed with the monitoring and controlling of processes. The process data is stored and the controlling agent constantly checks for anomalies for the criteria entered by the process manager.

3 Evaluation

Workflow systems are driven by the process models, which describe the workflow process. A sample process model for ordering a book is shown in figure 2. The activities associated with the process include order entry, inventory check, credit check, evaluation, approval, billing, shipping, archiving and the activity associated with writing a rejection letter. Evaluation of workflow systems are carried out on the basis of 20 workflow patterns as described by Van der Aalst[13]. We evaluate our system for these workflow patterns and also the communication patterns and explain the features of the agent framework, which provide support for these patterns.

3.1 Workflow Patterns

As we use coloured petri nets [16] as the process-modeling tool, we satisfy the basic workflow patterns such as sequence, parallel split, synchronization, exclusive choice and simple merge [13]. Table 1 shows the categorization of patterns and shows the level of support that our system provides. The notation "++" is used when the pattern is supported by the Petri net and "+" notation denotes that the pattern is not supported by the petri net but can be achieved by using our agent based framework. Van der Aalst categorizes workflow patterns into the following.

1. Advanced branching and synchronization patterns
2. Structural patterns
3. Patterns involving multiple instances
4. State based patterns
5. Cancellation patterns

Out of these categories of patterns, the coloured petri net formalism supports patterns described by categories 1, 2 and 4 [13].

3.1.1 Advanced branching and synchronization patterns

The patterns in this category include the multiple-choice pattern, synchronization merge pattern, multiple merge pattern and the discriminator pattern. All the four of the above-described patterns can be achieved by examining the colour of the token. The token can have attributes, which can be evaluated so that the transition could be fired and one of the possible paths (branching or merging) can be chosen.

3.1.2 Structural patterns

Structural patterns include *Aribitrary cycles pattern* which describes the point in a workflow process where one or more activities can be done repeatedly. This can be implemented in Petri nets.

Implicit termination pattern is that the given sub process should be terminated when there is nothing else to be done. Our framework supports implicit termination (refer to Case C of the example given in figure 4). The user can interact with the system to indicate the occurrence of an external event, which could trigger the termination of the process.

3.1.3 Patterns involving multiple instances (MI's)

These patterns are not directly supported through Petri nets[13]. But the same can be achieved by the combination of other patterns or through the agent framework. MI without synchronization is possible by using arbitrary cycles pattern. In this case instead of having multiple instances of a same activity, the process is repeated for certain number of cycles. Figure 3 shows a process (Case A), the MI described by Aalst (Case B) and the arbitrary cycles (Case C). It can be seen that the Case B described by Aalst can be modified into Case C, which uses the arbitrary cycles. This can be achieved by the arc expression, which would check the attribute of the token representing the number of times the task has to be repeated.

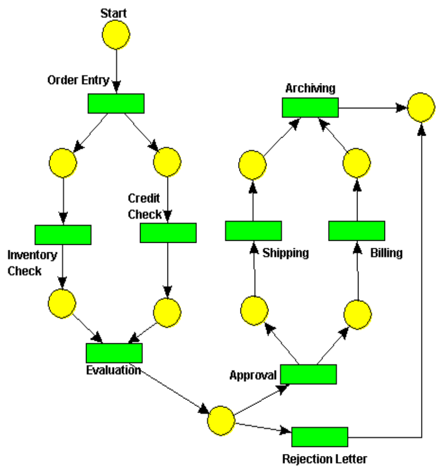

Figure 2: Process model of ordering a book

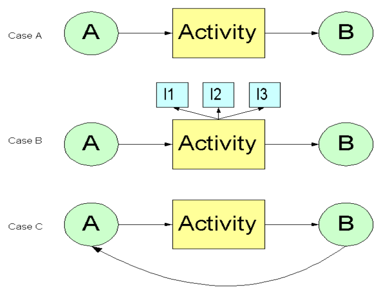

Figure 3: Showing processes with a single instance activity, multi instance activity and multi instance activity achieved through arbitrary cycles (iteration)

3.1.4 State based patterns

Deferred choice pattern describes the execution of one of the two alternatives paths. The choice, which path is to be executed should be determined by some environmental variables. Interleaved parallel routing pattern describes execution two activities in random order, but not in parallel. Milestone pattern enable an activity until a milestone is reached. These patterns are inherently supported by coloured Petri nets[13].

3.1.5 Cancellation patterns

Cancel activity pattern is cancellation or disabling of an activity. The CPN will not be able to cancel an activity because in CPN we only have local control around a

transition. Depending upon the value of tokens in the input places the tokens on the output place can be generated. But this can be achieved using the higher-level language support that executes the process model. The agent-based framework can provide a user interface such as a stop activity button so that the activity can be cancelled. This is possible since a every case is executed by a separate process agent.
Cancel case pattern is the cancellation of the entire case of a process. The same argument for the previous case holds good. The user interface of the agent framework can support it. Figure 4 shows how the framework can support the cancellation patterns. *Activity2* is the active activity. The user can now decide to cancel the activity or the whole case.

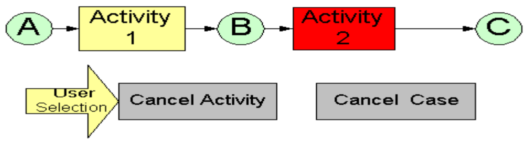

Figure 4: Showing the user interface for cancellation of running case/activity of the Petri net process model.

Categorization of workflow patterns	**Workflow Patterns**	**Support in JBees (++ or +)**
Basic Patterns	Sequence	++
	Parallel Split	++
	Synchronization	++
	Exclusive Choice	++
	Simple Merge	++
Advanced branching and synchronization patterns	Multi Choice	++
	Synchronizing Merge	++
	Multi Merge	++
	Discriminator	++
Structural patterns	Arbitrary Cycles	++
	Implicit Termination	+
Patterns involving multiple instances	MI without Synchronization	+
	MI with a Priori Design Time Knowledge	+
	MI with a Priori Runtime Knowledge	+
	MI without a Priori Runtime Knowledge	+
State based patterns	Deferred Choice	++
	Interleaved Parallel Routing	++
	Milestone	++
Cancellation patterns	Cancel Activity	+
	Cancel Case	+

Table 1: Showing the categorization of workflow patterns and their support in JBees.

Categorization of communication patterns	**Communication Patterns**	**Support in JBees (++ or +)**
Synchronous	Request/Reply	++
	One-way	++
	Synchronous Polling	++
Asynchronous	Message Passing	++
	Publish/Subscribe	+
	Broadcast	+

Table 2: Showing the categorization of communication patterns and their support in JBees.

3.2 Communication Patterns

Communication is realized by the exchange of messages between different processes. Our agent-based system is designed for sending and receiving messages based on the FIPA[6] protocol. In this section we evaluate JBees for the various communication patterns. An example (shown in 5) of the communication is how the sub processes are executed. To execute a sub process, the process agent of the parent process instantiates another process agent. The process related communication takes places between the parent process agent and the sub process agent. Table 2 shows the categorization of communication patterns and shows the level of support that our system provides. The notation "++" is used when the pattern is supported by FIPA specification that our framework is built upon and "+" notation denotes that the pattern is not supported by FIPA but can be achieved by using our framework.

3.2.1 Synchronous communication

Request/Reply pattern is the communication pattern in which the sender sends a request and waits for a reply. The communication scenario shown in Figure 5 is an example of this pattern. ***One-way pattern*** is the pattern where a sender makes a request to a receiver and does not wait for response. Our FIPA compliant framework supports these patterns. ***Synchronous polling pattern****:* The sender communicates a request to a receiver but instead of blocking it continues processing and constantly checks for response. If a resource is not available at a point of time, the process agent continuously keeps checking with the resource broker whether any resource is available after a fixed interval of time. Figure 6 shows the communication between the process agent and the resource broker agent.

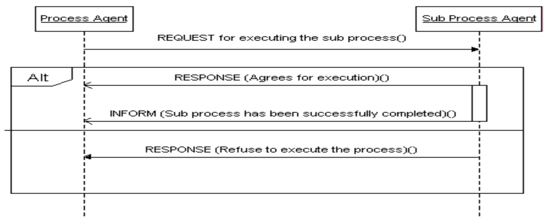
Process Agent
Sub Process Agent
REQUEST for executing the sub process()
Alt
RESPONSE (Agrees for execution)()
INFORM (Sub process has been successfully completed)()
RESPONSE (Refuse to execute the process)()

Figure 5: showing the communication between the parent process agent and the sub process agent executing the main and sub process respectively.

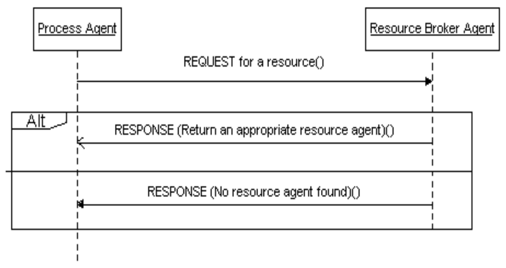
Process Agent
Resource Broker Agent
REQUEST for a resource()
Alt
RESPONSE (Return an appropriate resource agent)()
RESPONSE (No resource agent found)()

Figure 6: Showing the communication between the process agent executing a process model and the resource broker agent

3.2.2 Asynchronous communication

Message Passing Pattern is an asynchronous communication pattern in which the sender receives no response. When the request reaches the receiver it processes the message and performs appropriate actions. Though our FIPA compliant framework supports this form of communication, it is not used in the context of workflows as the feedback from agents about the starting and completion of tasks/activities should have reply/response. ***Publish/Subscribe pattern*** is the asynchronous communication pattern in which the sender sends the message to those who have already expressed their interest in receiving the messages when an event has occurred. This pattern is not supported by FIPA yet but can be implemented in the framework by maintaining the list of all agents that would express their interest in receiving certain kinds of messages. ***Broadcast Pattern*** is the form of communication in which all the participants receive a message. Though this has not been supported by the FIPA protocol, it can be achieved in our framework by sending messages to all agents that collaborate in a particular platform. The list of all collaborating agents can be obtained and the message can be sent to all agents individually.

3.3 Support for workflow and communication patterns in JBees

It can be observed form sections 3.1, 3.2 and figure 7 that sixty five percent of the workflow patterns are supported directly by Petri nets and the agent-based framework can support the rest of the patterns. We have also described the communication patterns that our system supports. Four out of the six communication patterns are supported directly by our agent-based framework and the other two can be supported with few changes.

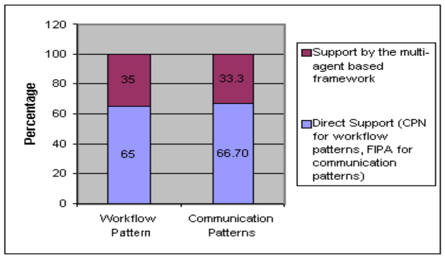

Figure 7: Graph showing the support for workflow and communication patterns in JBees

4 Conclusion

We have evaluated the capability of our workflow system both from process modeling point of view as well as the inter process communication viewpoint. Through these patterns, the CPN models executed by the multi-agents have addressed issues on flexible workflow systems by supporting distribution and adaptability of processes. We agree to the viewpoints of van der Aalst [14] that Petri nets could be considered as a standard for modeling workflows but they should be aided by multi-agents to provide the flexibilities that include adaptability and distribution of processes. Owing to the support of distributed and adaptive processes, workflow systems modeled using CPN's and managed by multi-agents have started emerging ([18] and [19]). Our described system is available under the GNU Lesser General Public License [3] on the internet [9].

5 Acknowledgements

The authors wish to acknowledge the work of Lars Ehrler and Martin Fleurke in the implementation of the agent based workflow system.

References

1. 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.
2. Martin Fleurke, Lars Ehrler, and Maryam Purvis, 'JBees - an adaptive and distributed framework for workflow systems', in Workshop on Collaboration Agents: Autonomous Agents for Collaborative Environments (COLA), Halifax, Canada, eds., Ali Ghorbani and Stephen Marsh, pp. 69–76, http://www.cs.unb.ca/˜ ghorbani/cola/proceedings/NRC-

46519.pdf, (2003). National Research Council Canada, Institute for Information Technology.

3. Free Software Foundation. GNU Lesser General Public License, 2000.
4. S. Meilin, Y. Guangxin, X. Yong, and W. Shangguang,‘ Workflow Management Systems: A Survey. ’, in Proceedings of IEEE International Conference on Communication Technology, (1998).
5. Mariusz Nowostawski. JFern – Java based Petri Net framework , 2003.
6. FIPA, FIPA Communicative Act Library - Specification. 2002. http://www.fipa.org/specs/fipa00037
7. Martin K. Purvis, Stephen Cranefield, Mariusz Nowostawski, and Dan Carter, ‘Opal: A multi-level infrastructure for agent-oriented software development’, The information science discussion paper series no 2002/01, Department of Information Science, University of Otago, Dunedin, New Zealand, (2002).
8. J.W. Shepherdson, S.G. Thompson, and B. Odgers, ‘Cross Organisational Workflow Coordinated by Software Agents’, in CEUR Workshop Proceedings No 17. Cross Organisational Workflow Management and Coordination, San Francisco, USA, (1998)
9. Department of Information Science University of Otago. JBees. http://jbees.sourceforge.net, 2004.
10. W.M.P van der Aalst, ‘ Exterminating the Dynamic Change Bug: A Concrete Approach to Support Workflow Change ’, Information Systems Frontiers, 3(3), 297–317, (2001).
11. W.M.P van der Aalst and K. van Hee, Workflow Management: Models, Methods, and Systems , MIT Press, 2002.
12. W.M.P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computer*s, 8(1):21–66, 1998.
13. W.M.P. van der Aalst and A.H.M. ter Hofstede Workflow Patterns: On the Expressive Power of (Petri-net-based) Workflow Languages. In: Kurt Jensen (Ed.): *Proc. of the Fourth International Workshop on Practical Use of Coloured Petri Nets and the CPN Tools, Aarhus, Denmark, August 28-30, 2002*, pages 1-20. Technical Report DAIMI PB-560, August 2002.
14. W.M.P. van der Aalst. Don’t go with the flow: Web services composition standards exposed, IEEE Intelligent Systems, Jan/Feb 2003.
15. P. Wohed, W.M.P. van der Aalst, M. Dumas, and A.H.M. Hofstede. Pattern Based Analysis of BPEL4WS. QUT Technical report, FIT-TR-2002-04, Queens-land University of Technology, Brisbane, 2002.
16. 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.
17. Savarimuthu, B.T.R., Purvis, M. and Fleurke, M. (2004). Monitoring and Controlling of a Multi-agent Based Workflow System. In Proc. Australasian Workshop on Data Mining and Web Intelligence (DMWI2004), Dunedin, New Zealand. CRPIT, **32**. Purvis, M., Ed. ACS. 127-132.
18. Vidal, J.M Buhler, P and Stahl, C (2004), Multi agent systems with workflows. IEEE computer society*,* Jan-Feb, *76-82*
19. K. Palacz and D.C. Marinescu. An agent-based workflow management system. In Proc. AAAI Spring Symposium Workshop "Bringing Knowledge to Business Processes", Standford University, CA
20. W.M.P. van der Aalst. Three Good reasons for Using a Petri-net based Workflow Management System. In S. Navathe and T. Wakayama, editors, Proceedings of the International Working Conference on Information and Process Integration in Enterprises (IPIC’96), pages 179– 201, Cambridge, Massachusetts, 1996.
21. The workflow management coalition. The workflow reference model, 1995.

22. J. Bradshaw. An Introduction to Software Agents . In J. Bradshaw, editor, Software Agents, pages 3–46. MIT Press, 1997.
23. M.J. Wooldridge. Intelligent Agents . In G. Weiss, editor, Multiagent Systems, pages 27–77. MIT Press, 1999.
24. Y. Shoham. An Overview of Agent-Oriented Programming. In J. Bradshaw, editor, Software Agents, pages 271–290. MIT Press, 1997.
25. K.P. Sycara. Multiagent Systems . AI magazine, 19(2):79–92.
26. Mariusz Nowostawski, Geoff Bush, Martin K.Purvis, and Stephen Cranefield. A Multilevel Approach and Infrastructure for Agent-Oriented Software Development. In International Work-shop on Infrastructure for Agents, MAS and Scalable MAS, http://www.umcs.maine.edu/˜wagner/workshop/01_nowostawski_bush_purvis_etal.pdf, 2001.
27. G. Joeris. Decentralized and Flexible Workflow Enactment Based on Task Coordination Agents . In 2nd Int'l. Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS 2000 @ CAiSE*00), Stockholm, Sweden, pages 41–62. iCue Publishing, Berlin, Germany.
28. N.R. Jennings, P. Faratin, T.J. Norman, P. O'Brien, and B. Odgers. Autonomous Agents for Business Process Management. Int. Journal of Applied Artificial Intelligence, 14(2): 145–189, 2000.
29. M.E. Nissen. Supply Chain Process and Agent Design for E-Commerce. In 33rd Hawaii International Conference on System Sciences, 2000.
30. M. Wang and H. Wang. Intelligent Agent Supported Flexible Workflow Monitoring System . In Advanced In-formation Systems Engineering: 14th International Conference, CAiSE 2002, Toronto, Canada, 2002.
31. H. Stormer. AWA - A flexible Agent-Workflow System . In Workshop on Agent-Based Approaches to B2B at the Fifth International Conference on Autonomous Agents (AGENTS 2001), Montreal, Canada, 2001.
32. Q. Chen, M. Hsu, U. Dayal, and M.L. Griss. Multi-agent cooperation, dynamic work ow and XML for e-commerce automation. In fourth international conference on Autonomous agents, Barcelona, Spain, 2000.
33. Purvis, M. K. and Purvis, M. A. and Lemalu, S., "A Framework for Distributed Workflow Systems", Proceedings of the Hawai`i International Conference on System Sciences (HICSS-34), (CD-ROM) IEEE Computer Society Press, Los Alamitos, CA (2001).