Journal of Information Systems and E-Business Management manuscript No.
(will be inserted by the editor)

Agent-based Workflow Management Systems (WfMSs)

JBees – a distributed and adaptive WfMS with monitoring and controlling capabilities

Lars Ehrler[1], Martin Fleurke[1,2], Maryam Purvis[1], Bastin Tony Roy Savarimuthu[1]

[1] Department of Information Science, University of Otago, Dunedin, New Zealand, e-mail: lars@larsehrler.de,{tehrany,tonyr}@infoscience.otago.ac.nz
[2] Department of Computer Science, University of Twente, Enschede, The Netherlands, e-mail: a.m.fleurke@student.utwente.nl

Received: date / Revised version: date

Abstract Workflow management systems (WfMS) based on agent technology can cope with the rapidly evolving business environment better than most other systems as they are more flexible and open. In this paper we describe a possible architecture of such a system by means of our prototype WfMS called JBees. The combination of collaborating agents and the Coloured Petri Net (CPN)-formalism in JBees enables a flexible and adaptive system with the possibility of simulation, analysis and monitoring of the process execution in order to identify potential inconsistencies and to provide appropriate information to the workflow administrator for the purpose of the process improvement.

Key words Workflow Management System – adaptability – software agents – workflow monitoring – Coloured Petri nets

1 Introduction

Workflow management systems (WfMSs) [31,14,21] 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, and
- Automated coordination and collaboration between various business entities.

Existing, commercially available workflow management systems 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 [22]. Improvements can be made by employing a distributed network of autonomous software agents that can adapt to changing circumstances.

In the past, WfMSs were used in more well-defined activities, such as manufacturing, where the processes tend to be more established and stable. But in the current climate WfMSs 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, 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 wanting an adaptive WfMS are as follows [4]:

- 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, new requirements may be imposed which can impact the process definition. This change in the market may also include the availability of some new technologies, which may require the modification of the process as well.

2 Background

2.1 Coloured Petri Nets

Coloured Petri nets (CPNs) are used to model workflow systems, due in part to their sound mathematical foundations and to the fact that they have been used

Send offprint requests to: Maryam Purvis
Correspondence to: Department of Information Science, University of Otago, PO Box 56, Dunedin, New Zealand, e-mail: tehrany@infoscience.otago.ac.nz

extensively for modelling distributed systems with concurrent activities [12]. Coloured Petri nets consist of the following basic elements:

- Tokens, which are typed markers with values—in our implementation the type can 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. Transitions 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 multisets of token values affects the enablement and firing of transitions.

Some reasons for preferring Petri net modelling to other notations used for workflow modelling are given by [3]:

- 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) [26]
- 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 modeller to start with a coarse model of the process and gradually refine each of the sub-processes into a separate Petri net model.

2.2 Agent systems

Some commonly accepted characteristics of an agent are listed by Bradshaw [5]: reactivity, autonomy, collaborative behaviour, adaptivity and mobility. The most important attribute is probably autonomy. Wooldridge [33] (and similarly Shoham [23]) discusses the difference between agents and objects. They both mention three main distinctions, which are all related to the autonomy of agents: Agents have control over their behaviour, agents have flexible behaviour (they can behave reactively or proactively) and an agent often has its own thread(s) of control.
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. 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 [19], developed at the University of Otago since 2000. It meets the standards of the Foundation for Intelligent Physical Agents (FIPA) [1] for agent platforms and incorporates a modular approach to agent development [18].

2.3 Former approaches to WfMSs

2.3.1 WfMSs using software agents In the context of WfMSs, agent technology has been used in different ways [13]. 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 [11,16]. 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 [24,32]. 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 [6], where an agent-based WfMS is used in conjunction with specialized agents that provide appropriate application-related services.

2.3.2 Adaptive WfMS Adaptive workflows have been discussed for many years and many people have described what should be done. 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. Transferring running work cases to a new model is still a difficult issue. This is indicated in a comparison of current WfMS that was done by Van der Aalst et al. [30].
The approach in much of the research is to define a limited set of possible transfers, such as inserting an extra task in sequence, skipping a task, or replacing a task with a new subnet. In this way the semantics of the workflow stay well defined. The drawback of this approach is that the set of possible transfers is very limited and/or the expressive power of the specification language is too

restricted, limiting the possibilities for specifying a process. For example the approach of Van der Aalst et al. [29] is based on inheritance, but it requires that every workflow should be derived according to a limited set of transitions from some basic workflow definition.

2.3.3 Monitoring and feedback Monitoring and feedback mechanism of workflow systems have been mentioned by researchers for many years. Few researchers have discussed the issues associated with monitoring and feedback ([7],[15]). When it comes to agent-based monitoring, there has been one proposed system [32], but this lacks feedback of the process model using agents and also does not cater for distributed monitoring, which is central to any workflow system as described by Van der Aalst et al. [30].

3 JBees

3.1 The architecture

Our research is focussed 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 [19] and uses the CPN execution tool JFern [17]. A first description of JBees can be found in our former paper [9]. 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.

3.1.1 Management Agent. The manager agent provides the user interface for the human workflow manager. It can:

- Create and delete role definitions and process definitions
- Instantiate a new process instance
- Create resource agents for new resources
- Simulate the execution of a process

3.1.2 Storage Agent. A storage agent manages the persistent data, for instance the definitions of tasks, roles and processes and the monitored data. It also notifies all management agents if the data has changed (for example one management agent adds a definition, and the storage agent notifies all other management agents that there is a new definition).

3.1.3 Process Agent. A process agent is responsible for the execution of one particular case. For each work case and for each sub work case a new process agent is created. The CPN model is provided by the management agent or by the "parent" process agent and the process agent uses JFern to execute this model.

3.1.4 Resource Agent. The resource agent is the user interface for the human resource or the interface for some tool which can do tasks automatically(such as printers and scanners). Every resource has its own resource agent. The agent represents the resource in the system and negotiates the resource allocation on behalf of the resource and exchanges the necessary information for the execution of a task.

3.1.5 Resource Broker Agent. The resource broker is responsible for the resource management. Every resource agent is registered with at least one of the resource brokers and a process agent requests the resource broker to identify and allocate a suitable resource. Different strategies for the resource management can be incorporated just by replacing the agent with a new agent, that employs a new strategy.

3.1.6 Monitor Agent. The monitor agent gathers the data in the system that is necessary to analyze workflows, such as execution times and resource utilization. The monitor agent gathers this data associated with a particular case and (after the case is finished) sends this data to the storage agent for persistent storage.

3.1.7 Control Agent. The control agent provides the feedback mechanism required for process re-engineering. The control agent continuously senses the anomalies or violations of the criteria specified by the manager and sends warning messages to the manager agent and also logs those messages. The manager agent or the human manager decide the appropriate corrective measures that has to be carried out.

3.2 Advantages of using an agent enhanced WfMS

We outlined in section 2.2 that the use of software agents facilitates the design of a distributed and open platform because agents are loosely coupled components forming an open system. New technologies and techniques can easily be incorporated by introducing new specialized agents into the system.
The use of such an open platform has advantages for building a workflow management system. Firstly it facilitates cooperation with other organizations. The essential pieces for interoperability are ontologies and interaction protocols. If the organizations agree on standard ontologies and protocols, they can easily cooperate even if they have completely different platforms and systems. Secondly, the use of software agents gives us the opportunity to have a flexible system. For example the resource management strategy can change dynamically (by just starting an agent with a new strategy), resources can register or de-register, and process definitions can be changed more easily.

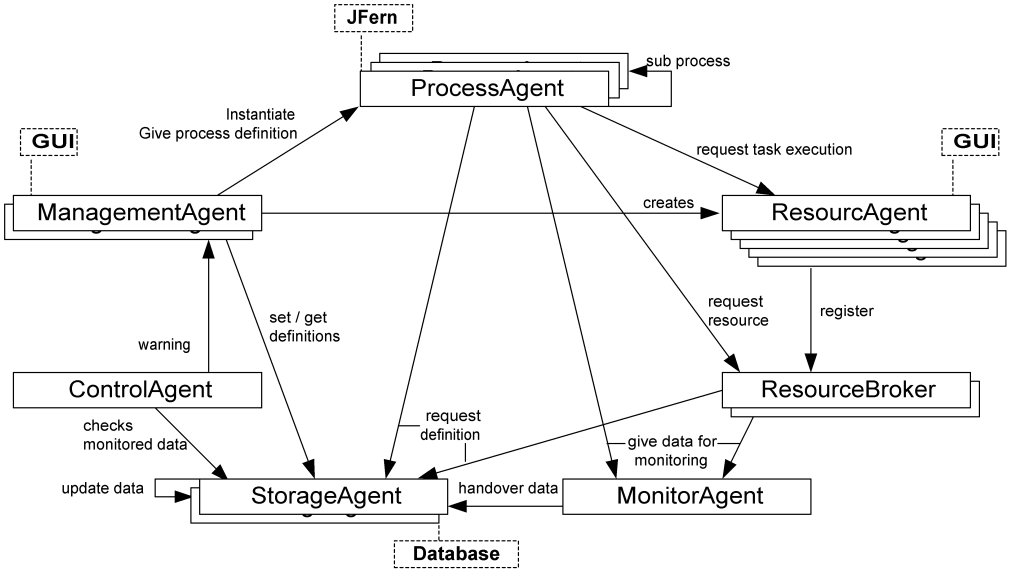

Fig. 1 An overview over the JBees architecture

This flexibility provides adequate support in implementing an adaptive and distributed system. New process models (when they are required) can be dynamically incorporated into the system (see section 3.5) and each agent can be located in a separate host on the Internet (see section 3.7).

3.3 Execution of Workflows

In order to process a new work case, the management agent creates a new process agent providing the definition of the process. The process agent starts the execution of this case by invoking JFern.
The process agent requests all the definitions of the tasks from the storage agent (the process definition contains only the names of the task – the definitions themselves are not stored in the CPN model so that they can be easily reused in different process definitions). Every task definition contains all roles that can execute this task. These roles are needed to request an appropriate resource from the resource broker (see section 3.4). The process agent contacts the resource allocated by the resource broker directly to hand over the information necessary for executing the task.

3.4 Resource allocation

The workflow is described by a coloured Petri net. Normally, a transition representing a task is enabled if there is a job token that specifies the context of the current job (work case instance), and a resource token that indicates that there is a resource to do the task. The resources are taken from a CPN model place that holds all resources in the system. To provide more flexibility the resource required for each transition is not included in the model.Since the resources are decoupled from the model, they can be easily changed. In this approach resources have to be explicitly requested when required by a transition. This process (obtaining the required resources) is represented in another subnet not shown in the process model to avoid cluttering the net with the modelling elements that are not central to the model.
The request for a resource contains the roles which can execute the task. The resource broker sends a request message to the appropriate resource agent which is capable of performing the task. The resource agent can either decline or accept the request. If the broker can successfully allocate a resource, it will return the name of chosen resource to the process agent.

3.5 Adaptability

As was indicated earlier, 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 can be a need to modify the CPN model (due to a 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 accomplish this task, we use an

algorithm that calculates the region in which transfers are unsafe [8]. The algorithm is an improved version of Van der Aalst's algorithm [28] to calculate the minimal change region.

In order to determine whether a transfer of the case is safe or not, the algorithm identifies the region of the process model that is impacted by the change. The impacted region is identified by first calculating the difference between the old and new net. Second, any other net elements that relate to the affected elements via parallelism are included. Together these elements form the critical region that is examined in order to determine if the transfer is safe or not.
If the job token of the running case is outside of the corresponding region of the net, then the state is transferred to the proposed process model. Otherwise the transfer is not safe and requires a human manager to choose one of the following exception handling methods:

- Continue processing the instance according to the old definition
- Restart the instance with the new definition
- Delay the change until the job token gets during the workflow execution in a safe region and change then the process model
- Have the user manually transfer the token

A possible transfer is facilitated by our use of one agent for each work case. Therefore the work cases are executed independent of each other and a transfer from one case does not interfere with other work cases.

For more details concerning the algorithm used to identify this region refer to the Masters degree thesis by Martin Fleurke [8].

3.6 Reliability

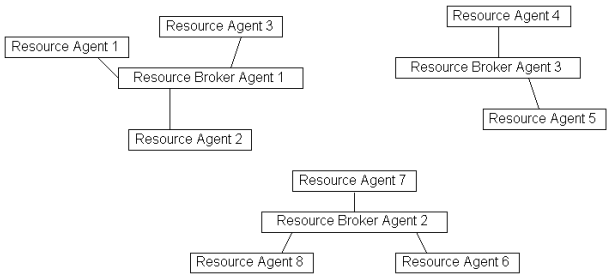

Fig. 2 Distributed resource management

One significant problem in the design of a distributed system is reliability. In particular, certain elements that are crucial to the execution of workflows might be prone to "central points of failure" (for example if they cannot be accessed or they fail themselves). A WfMS usually has two critical activities: the management of resources and the management of persistent data. If these elements are centralized and they fail, the whole system cannot function any more.
Our architecture based on agent technology provides an easy and flexible solution for this problem. A resource broker agent is responsible for the management of resources, but there can be several resource brokers. As you can see in Figure 2, every resource agent is registered with atleast one of the broker agents. If a resource broker fails or is not available, there are still other brokers which might be available and one of them can be contacted.Possibly a resource that is not ideal will be allocated in this case, however, the process will still be executed. The management of persistent data is managed in a similar way.

3.7 Distribution

An organization with offices/departments at different locations would require distributed workflow models. As shown in section 2.2, the adoption of agent technology facilitates the support for these distributed workflow modelling and execution.The mechanism of distribution in JBees has been carried out in the following ways:

1. The process model can be distributed. A model can be separated into several sub-processes which can be located on different hosts as they are executed by several process agents. The sub-processes could span various geographical locations. For example a shipping sub-process can be located in the shipping department while the billing sub-process can be located in the finance department of an organization — both departments may be in different cities or even countries.
2. The resource management can be distributed. Even if the process is executed in one location, resource brokers at several locations can be asked for a resource. Therefore the execution of a task can be delegated to a remote resource with possibly specialized skills, even though the process itself is located at a completely different location.
3. JBees incorporates an agent-based distributed database. A central database can become a bottleneck or, worse, a central point of failure in a system. Therefore we incorporated a distributed database, managed by agents. The system can have several storage agents — each agent with its own local database(a node of the distributed database) underneath. Data can be retreived/stored by all agents through any of the storage agents. When any of the storage agents senses the change of data, it synchronizes the distributed database by informing all other storage agents to update the data.

3.8 Simulation

In the process of creating new process models these models needs to be tested and evaluated for their efficiency. For this purpose JBees has a special simulation agent(a dedicated process agent). The human manager can choose several parameters such as number of resources, number of cases or delay between actions and the agent simulates the execution of the process. The data collected by the simulation agent can be analyzed and also used for improving the process model.

3.9 Monitoring

Monitoring is an indispensable part of any WfMS. Every case that is executed in a process model has to be monitored for its various properties such as time taken to complete the process, the various resources employed, time taken by the resources to complete the tasks and waiting times of the jobs in the queue to be served by a resource. Our initial work on monitoring can be found in the previously published paper[20].
Our architecture incorporates the modules to examine, analyse and display the properties of the workflow system. This data is stored after simulation (as described in section 3.8) and also after enactment of every case.
Due to the distributed nature of the system we need new strategies for monitoring the system in enactment mode. The information is distributed and work cases can have modified process definitions, several sub work cases and several agents associated with the case, which are independently working. Our approach is to have a dedicated monitor agent collecting all information. Information which belongs to the same case will be collected and integrated by this agent. This data can be used to improve the workflow while executing (such as employing more resources of a role, that is in demand).
In both cases (simulation and enactment), the data is sent to the storage agent to store this data persistently. The human manager of the WfMS can choose any of the properties to analyse the state of that property at any particular point of time. The data obtained from simulation and enactment can be used for drawing graphs for the purpose of monitoring. We have integrated the JFreeChart, an open source Java API [2] with our framework to draw and display graphs using the data collected by our system.

3.10 Feedback mechanism

During the enactment and also while simulating various scenarios, the user/manager of the workflow system specifies certain criteria that have to be looked into continuously and should be constantly compared with the data obtained from the simulation or enactment. The user could specify whether the criteria holds for specific processes or for all processes. The criteria used in the feedback mechanism are implemented using database queries. The manager constructs these queries depending on which criteria he/she needs to monitor and also the frequency at which the control agent should report violations of these criteria.
When the controlling agent senses anomalies or violations of these criteria it sends a warning message to the management agent. An example of these criteria could be the overall completion time for any given case not exceeding more than a given value or the resource utilization not decreasing below a certain value. The controlling agent also logs the messages to the warning log so that the human manager can handle that particular problem if he/she is not available at that time. The aim of this design is to capture all possible critical conditions for continuous optimization of the processes.

4 Example

4.1 Simple Example

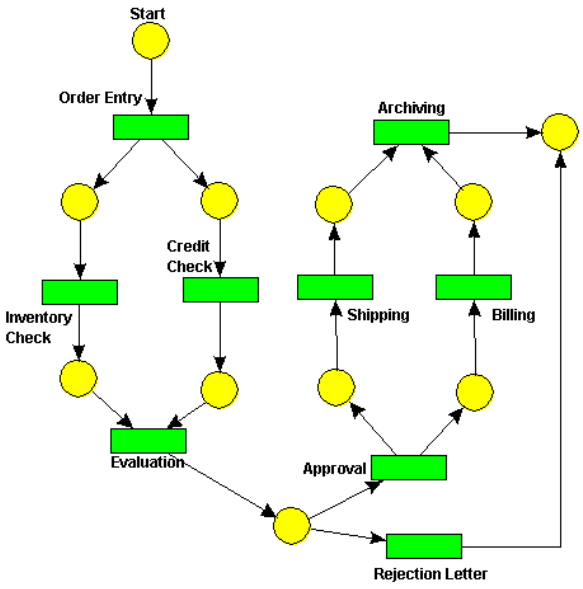

Fig. 3 The CPN of the example workflow

To illustrate the functionality of the system, we chose a simple process of ordering a book. The process model of this is shown in Figure 3.
After the customer orders a book, the inventory has to be checked whether a copy of this book is there and the credit rating of the customer has to be checked. These checks are done in parallel to speed up the process and afterwards the results are evaluated. Based on the evaluation it is decided whether the order can be processed or should be rejected. 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 so that one can handle possible customer complaints.

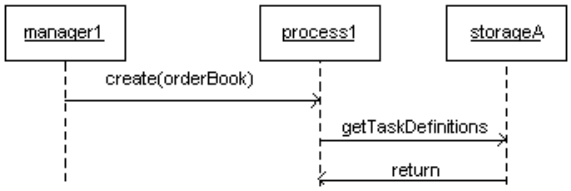

Fig. 4 The sequence diagram for starting a new work case

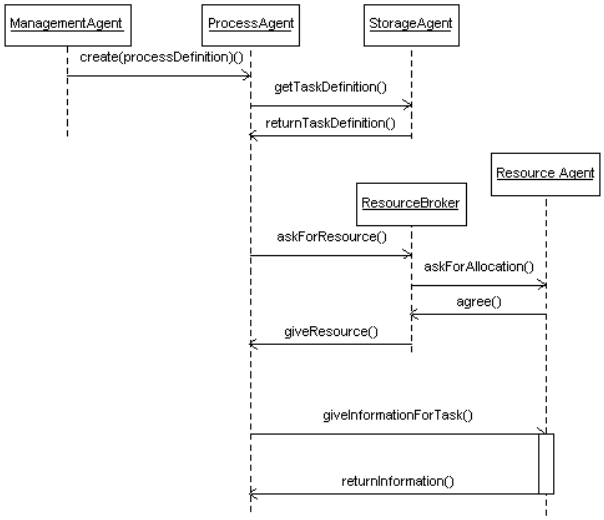

Fig. 5 The sequence diagram for allocating a resource and executing a task

Suppose a new order arrives. The first step of creating a new process agent and getting all task definitions is shown in Figure 4.
The process agent "process1" starts executing the work case by putting a job token in the place called Start(shown in Figure 3). This activates the transition Order Entry. According to the task definition, a resource of the role processor is needed. The process agent asks the resource broker for a resource of this role. The broker will allocate a resource of this type and return the name of this resource to the process agent. In case it cannot allocate a resource it will send a message to the process agent that it failed to allocate a resource. Then, the process agent has to decide how to handle this scenario(for instance by asking another resource broker, or waiting a certain time and asking the same resource broker again).

After the resource executes the task and returns the result of the task, the JFern engine gets notified that the task Order Entry has been successfully executed and it continues executing the CPN model. The process of allocating a resource and executing a task is shown in Figure 5 and this process is repeated for every task.

4.2 Demonstration of Adaptability

Suppose the company changes the way an order is dealt with. The process model in the right of Figure 6 is the new definition of this workflow. The tasks shipping and billing are now executed in series, which changes the degree of parallelism.
Our architecture, which assigns one process agent to every work case, gives the flexibility to handle this change:

- New work cases start with the new definition.
- Running work cases are handled individually. This means every single process agent decides whether the work case is in a safe region or not. If the token is in the safe region, the agent changes the definition to the new process model as shown on the right of Figure 6. In our example a safe work case would be for a situation in which the Approval transition has not been fired (in other words, there should be no token in the dashed region shown in the diagram on the left of Figure 6). If Approval has fired, there appears to be no automated algorithm that can decide how to transfer the case.In this case, the human manager can decide what to do with unsafe work cases: either to go on with the old definition or allow manual transfer of the token by the human manager.

The use of one agent for every work case allows us to easily change a process only for a specific number of particular work cases. The manager has only to select the agents that are running these work cases and give them a new definition.

4.3 Demonstration of Distribution

Let us now assume that the company has several branches, for example suppose the shipping department is located in Dunedin and the customer service is located in Auckland. Because of this distributed nature of the company, it is desirable to have a hierarchical workflow and run the sub-processes at the local hosts.
The top-level CPN model for this example is shown in Figure 7. The subnets are well defined. Now the process agent gets this top level CPN model and executes it. Every transition has an attribute that indicates if it is a sub-process, a task, or a control transition. Depending on the value of this attribute, the process agent executes the action code of the control transition, requests a resource for the task or creates a new process agent for the sub-process. In case of a new sub-process it starts a

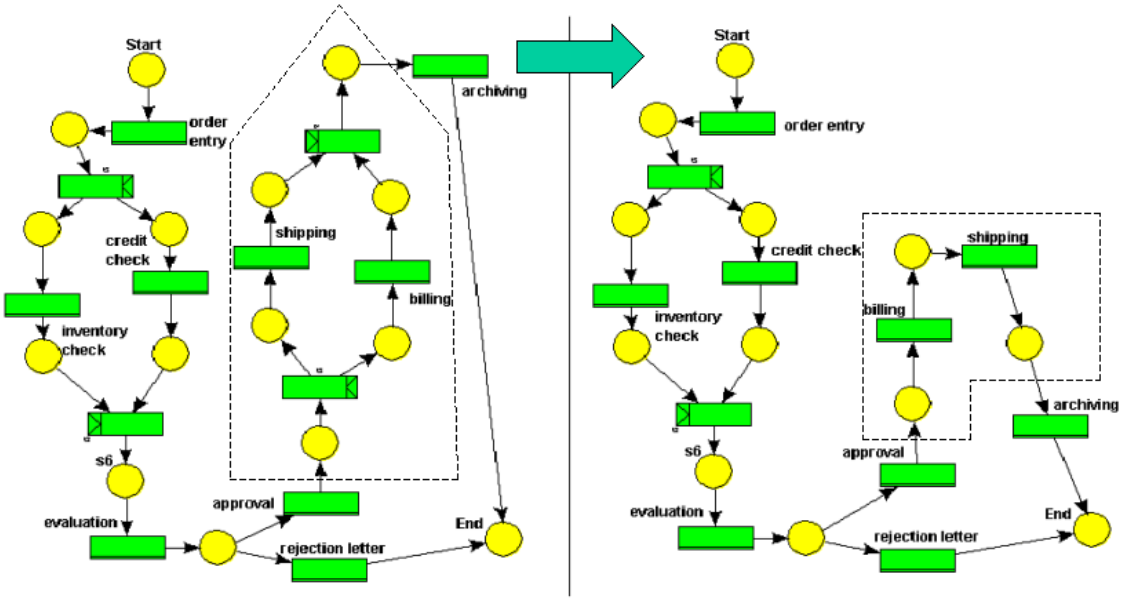

Fig. 6 The old and the new nets showing the unsafe region

new process agent (which can be located at another host if necessary). This process of the distributed workflow hierarchy can be arbitrarily extended. Note that there is no single Petri net that controls the entire process. There is one agent that controls the top-level process, but the agents for the sub-processes work independently. The results of the sub-processes are integrated accordingly with the top-level process. Sub-processes are only created when they have to be executed, so they automatically have the latest versions of the sub-process definition.

4.4 Monitoring

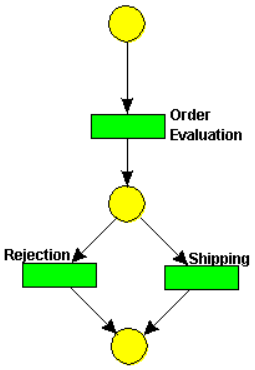

Fig. 7 The overall process of the distributed workflow

In simulation enactment mode, the various tasks of the process model were associated with a set of values for the process parameters.These parameters include the number of resources, their availability and their ability (time taken) to do these tasks to simulate various "real life" scenarios. By varying these process parameters various test scenarios can be examined. The storage agent stores the results of the simulated cases.
In execution mode, every time an action is started or finished, the process agent sends a message with the exact time of these events to the monitor agent. This is also done by the resource broker in the event of resource registration or deregistration. The monitor agent collects these data. After it gets the notification message from the process agent that the work case is finished, it combines the data and sends it to the storage agent for persistent storage.

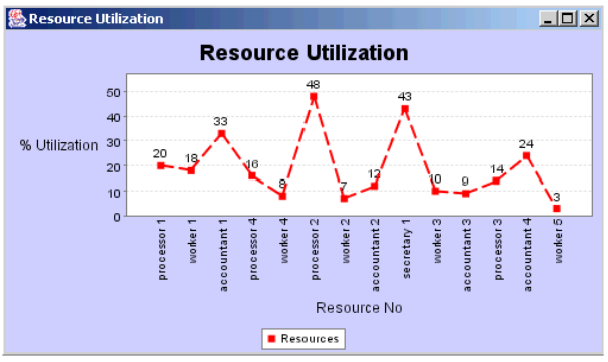

Fig. 8 The percentage utilization of the resources

Figure 8 shows the overall utilization of the resources and figure 9 shows the waiting time for various tasks for similar cases.

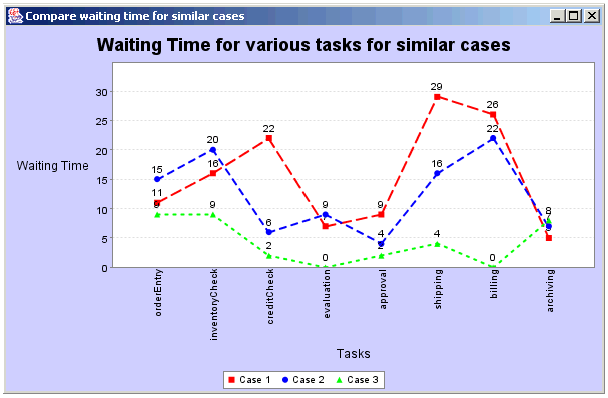

Fig. 9 The waiting time for various tasks for similar cases

4.5 The feedback mechanism

The human manager specifies the various criteria that are to be monitored continuously for violations. For example the criteria could be "The completion time for a case of process X should not exceed twice the average of completion times of all cases of the same process (process X)". The controlling agent looks for anomalies to these criteria and reports them to the manager agent. The manager agent might then decide to initiate the appropriate feedback action or display the warning so that the human manager can take appropriate action. This feedback mechanism helps in continuous process improvement.

5 Conclusion

In this paper we have described the architecture of a prototype agent-based workflow management system JBees. The use of coloured Petri nets as process formalism and of collaborating software agents as the basis of the systems provides more flexibility than existing WfMSs. JBees provides a high degree of distribution and can deal with various levels of adaptability. We implemented the described architecture and this paper showed solutions for the problems arising while implementing this system. Our existing framework has been endowed with monitoring and feedback mechanisms so that the various processes and cases can be studied, analyzed and the required feedback can be given to the workflow manager.

As part of our future work we intend to integrate into the currently implemented framework one of the existing CPN tools that supports formal analysis so that we can examine the model for certain properties such as reachability. This information as well as the monitoring information can be used to improve the processes in order to optimize the effectiveness of the system.
Another part of our future work will be the integration of sophisticated resource management. One step would be to store all tasks that have been performed by a particular resource (a task history) to allow more efficient resource scheduling. There are lots of resource management strategies already developed which we could as well apply to our system. The prototype described is available under the GNU Lesser General Public License [10] on the internet.[27]

Acknowledgements The authors wish to thank Mariusz Nowostawski for his help in implementing the system and improving JFern, and Prof. Martin Purvis for supporting this work.
This work has been supported by the Otago Research Grant ORG-0103-0304.

References

1. Foundation for Intelligent Physical Agents .
2. Object Refinery Ltd. JFreeChart . http://www.jfree.org.
3. 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, Camebridge, Massachusetts, 1996.
4. S. Bandinelli, A. Fuggetta, and C. Ghezzi. Process Model Evolution in the SPADE Environment, Technical Report No. 14, ESPRIT-III Project GOODSTEP (6115) . *IEEE Transactions on Software Engineering*, 19(12):1128–1144, 1993.
5. J. Bradshaw. An Introduction to Software Agents . In J. Bradshaw, editor, *Software Agents*, pages 3–46. MIT Press, 1997.
6. Q. Chen, M. Hsu, U. Dayal, and M.L. Griss. Multi-agent cooperation, dynamic workflow and XML for e-commerce automation . In *fourth international conference on Autonomous agents, Barcelona, Spain*, 2000.
7. B. Cui, Z. Odgers and M. Schroeder. An In-Service agent monitoring and analysis system. In *11th IEEE International Conference on Tools with Artificial Intelligence, Chicago, USA*, pages 237–244, 1998.
8. 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.
9. Martin Fleurke, Lars Ehrler, and Maryam Purvis. JBees - an adaptive and distributed framework for workflow systems. In Ali Ghorbani and Stephen Marsh, editors, *Workshop on Collaboration Agents: Autonomous Agents for Collaborative Environments (COLA), Halifax, Canada*, pages 69–76, http://www.cs.unb.ca/˜ ghorbani/cola/proceedings/NRC-46519.pdf, 2003. National Research Council Canada, Institute for Information Technology.
10. Free Software Foundation. GNU Lesser General Public License, 2000.
11. 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.
12. K. Jensen. *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use, Volume 1: Basic Concepts* . EATCS Monographs on Theoretical Computer Science. Springer-Verlag, 1992.
13. 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.
14. S. Meilin, Y. Guangxin, X. Yong, and W. Shangguang. Workflow Management Systems: A Survey. . In *Proceedings of IEEE International Conference on Communication Technology*, cscw.cs.tsinghua.edu.cn/cscwpapers/ygxin/WfMSSurvey.pdf, 1998.
15. M.Z. Muehlen and M. Rosemann. Workflow- based Process Monitoring and Controlling - Technical and Organizational Issues. In *33rd Hawaii International Conference on System Sciences, Maui, HI, USA*, 2000.
16. M.E. Nissen. Supply Chain Process and Agent Design for E-Commerce . In *33rd Hawaii International Conference on System Sciences*, 2000.
17. Mariusz Nowostawski. JFern - Java-based Petri Net framework , 2003.
18. Mariusz Nowostawski, Geoff Bush, Martin K. Purvis, and Stephen Cranefield. A Multilevel Approach and Infrastructure for Agent-Oriented Software Development. In *International Workshop on Infrastructure for Agents, MAS and Scalable MAS*, http://www.umcs.maine.edu/~ wagner/workshop/01_nowostawski_bush_purvis_et_al.pdf, 2001.
19. 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.
20. Bastin Tony Roy Savarimuthu, Maryam Purvis, and Martin Fleurke. Monitoring and controlling of a workflow management system. In *In Proc. Australasian Workshop on Data Mining and Web Intelligence (DMWI2004)*, 2004.
21. T. Schael. *Workflow Management Systems for Process Organisations* , volume 1096 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.
22. 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.
23. Y. Shoham. An Overview of Agent-Oriented Programming . In J. Bradshaw, editor, *Software Agents*, pages 271–290. MIT Press, 1997.
24. 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), Montral, Canada*, 2001.
25. K.P. Sycara. Multiagent Systems . *AI magazine*, 19(2):79–92.
26. The workflow management coalition. The workflow reference model, 1995.
27. Department of Information Science University of Otago. JBees. http://jbees.sourceforge.net, 2004.
28. 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.
29. W.M.P. van der Aalst, T. Basten, H.M.W. Verbeek, P.A.C. Verkoulen, and M. Voorhoeve. Adaptive Workflow: An Approach Based on Inheritance . In M. Ibrahim and B. Drabble, editors, *IJCAI'99 Workshop on Intelligent Workflow and Process Management: The New Frontier for AI in Business*, pages 36–45, 1999.
30. W.M.P. van der Aalst, A.H.M.t. Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns . Technical report, Queensland University of Technology, Brisbane, Australia, http://tmitwww.tm.tue.nl/research/patterns/, 2002.
31. W.M.P van der Aalst and K. van Hee. *Workflow Management: Models, Methods, and Systems* . MIT Press, 2002.
32. M. Wang and H. Wang. Intelligent Agent Supported Flexible Workflow Monitoring System . In *Advanced Information Systems Engineering: 14th International Conference, CAiSE 2002, Toronto, Canada*, 2002.
33. M.J. Wooldridge. Intelligent Agents . In G. Weiss, editor, *Multiagent Systems*, pages 27–77. MIT Press, 1999.