Different Perspectives on Modeling Workflows in an Agent Based Workflow Management System

Bastin Tony Roy Savarimuthu, Maryam Purvis, and Martin Purvis

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

Abstract. Most workflows are modeled from the point of view that the entire workflow can be perceived, modeled and executed as a single entity. While this is true in scenarios involving manufacturing industries it is not the same in e-business scenarios, which would involve various participants such as buyers, sellers, etc. In these scenarios it is difficult to explicitly model and use a single process model (with hierarchical sub processes). In these cases, a model of the interactions can be specified using Interaction Protocols (IPs). In this paper we discuss how our agent based workflow system supports process specific workflow models and the workflows based on interaction protocol models. The process modeler has the option of choosing one of these approaches of modeling or a combination of these approaches to model a workflow.

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 [3-5]. 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, WfMS 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 [10, 11] describe the advantages of our agent-based framework JBees [9], such as distribution, flexibility and ability to dynamically incorporate a new process model.

The process models that a workflow modeler designs for our system consists of the entire workflow model and the resources that are needed to perform various tasks associated with that model. While this is the typical scenario in most of the production oriented workflows, there are certain workflow scenarios in which Interaction Protocol (IP) models would be suitable than a overall workflow model. In this paper, we discuss how various agents that participate in an e-business scenario can use Interaction Protocols. The workflow modeler has the option to choose from two approaches of modeling, the process centric workflow or the workflow based on interaction protocols. The paper is organized as follows. A brief background of our work is given in Section 2. Section 3 describes how process centric workflows are used in our workflow system. In the Section 4 we explain how our agent-based architecture can be used for design and execution of interaction protocols. Section 5 provides some recommendations for choosing one of the mechanisms to model processes. The concluding remarks are presented in Section 6.

2 Background

2.1 Coloured Petri Nets (CPNs)

We use CPN as a formalism to model workflows in our system. The sound mathematical foundation behind the Coloured Petri nets (CPNs) makes it a very useful tool for modeling distributed systems. Petri nets consist of four basic elements namely *tokens, places, transitions* and *arcs*. A detailed description of CPNs can be found in [2].

2.2 Interaction Protocols (IPs)

Interaction protocols [1] are the specifications that allow a certain kind of conversation or exchange of messages between two agents. This reduces the search space of possible responses to an agent message. These interaction protocols can be used to model workflow scenarios in which agents are used to execute the process models and send messages to appropriate agents to perform a particular task. Each agent has the interaction protocol describing how to communicate with the other agent and what actions to perform when a new message arrives.

2.3 Existing Architecture

Our research is focused on developing an agent-based 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 [7]. JBees is based on Opal [8] and uses the CPN execution tool JFern [6]. The processes are modeled using CPNs [2]. A first description of JBees can be found in our previous papers [9-11]. Our enhanced system consists of seven Opal agents, which provide the functionality to control the workflow. 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 a feedback mechanism.

3 Mechanism 1: Demonstrating the Process Perspective Models Using an Example

Most workflow systems model application based on a single sequence of workflow process model such as fault processing system or diamond processing system. Our architecture described in the previous section is suitable for modeling from this perspective.

Figure 1 shows an order entry process for purchasing a book designed from a process perspective. The tasks include order entry, inventory check, credit check, evaluation, approval, billing, shipping, archiving and the task associated with writing a rejection letter. A task can be represented as a sub process and linked to the main process model forming a hierarchy of process models. Each task will be assigned to an appropriate resource agent. More details on how our system supports the process perspective can be found in our previous works [9-11].

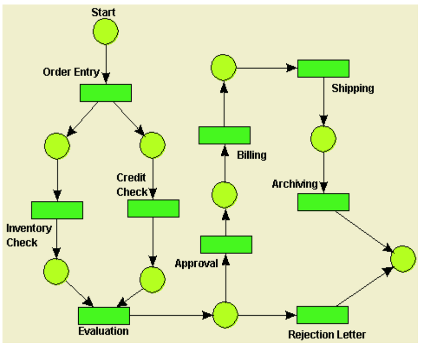

Fig. 1. The Order Entry process

4 Mechanism 2: Using Interaction Protocols (IPs) to Model Workflows

The reasons for using IPs for agents are manifold [1]. Interaction protocols are better intuitive models of how agents will interact than the message based communication between two agents. This conversation can be modelled as the interaction protocol between agents using Colored Petri nets. The advantage is of using an IP is that an agent is aware of the overall model. Some of the error handling mechanisms and global data handling mechanisms could be assigned to the agent. This gives a clear picture of various conversations an agent could be involved in.

Conversation structures are separated from the actions that are taken when an agent is involved in a conversation, facilitating the reuse of conversations in multiple contexts. The transitions that represent the actions can be implemented accordingly (by static binding or dynamic binding) depending upon the requirements.

Figure 2 shows the interaction protocols that each agent executes in a car insurance claim scenario. The car insurance workflow is expressed using the interaction protocols. The agents involved in the workflow scenario are the customer agent, insurance company manager agent (ICManager agent) and the panel beater agent. The customer captures different images of the damaged car and sends the photograph to the manager of insurance company. The insurance company manager forwards the photographs to the panel beater to get an estimate of the damage. The panel beater assesses the damage and sends a report to the insurance company manager. The insurance manager decides the insurance amount that has to be paid to the customer and informs the customer.

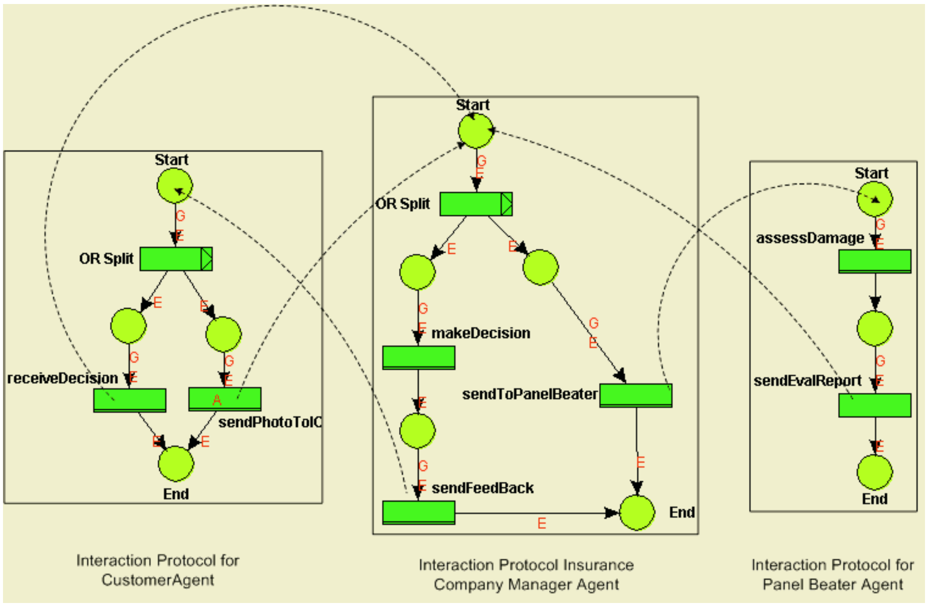

Fig. 2. Interaction protocols for car insurance claim scenario

4.1 Architectural Support for Interaction Protocols

Each agent in the scenario described above is an instance of a process agent. The process agent is capable of executing process models. In this case, we have three process agents that are capable of executing interaction protocols. The dotted arrows that emanate from the transitions in the interaction protocols show that the agent that executes the transition sends the messages. The arrowheads point to the start of an interaction protocol. This interaction protocol is executed when a message reaches a particular agent. For example when the insurance manager receives the images of the damaged car from the customer, the *ICManager* agent sends an email to the panel beater agent. When the message reaches the panel beater agent, it prompts the user of the system (the actual panel beater), to read the email. Then, the panel beater agent executes the interaction protocol. After the damage has been assessed, the agent sends a report to the *ICManager* agent. The sample code for sending this message encoded in the *sendToPanelBeater* transition is given below

```
JobToken jt = (JobToken) get (“X”);
if(jt.getAttribute(CarInsuranceOntology.TO_PANEL_BEATER))
{ sendMessageToPanelBeater();}
```

Before the execution of any protocol, a job token (assume, X) is created. This job token consists of certain attributes that enable the firing of various transitions. The attributes are stored in name and value pairs in a map. For the transition named *sendToPanelBeater* to fire, the job token should have a value corresponding to the name TO_PANEL_BEATER in the car insurance ontology. In this simple scenario, there has not been a need for additional resource agents to perform certain tasks. But more complex examples might need resource agents similar to the ones described in Section 3. For simplicity reasons, the not understood performative has not been shown in all the interaction protocols in figure 2. Our system also supports the dynamic changing of interaction protocols similar to the changing of process centric models.

5 Choosing Process Models

From workflow perspective, it is desirable to choose process centric way of process modeling. But form an agent perspective it might be desirable to have more autonomy to the agent by using the interaction protocol models for each agent. We feel that though both these perspectives are correct, the nature of the process that is being dealt with should be of importance when one thinks about choosing which way to go. In an application like fault processing system, the entire workflow is well defined and in this case, process centric approach would be easier to design and execute. But for fluid processes which might involve many alternative interactions between agents and that involves multiple agents interaction protocols need to be used. The car insurance workflow described in the previous section uses the interaction protocols. But the same workflow could also be modelled as a process centric model shown in figure 3.

The differences between the two approaches are given below.

a) Process centric approach is the widely used approach in workflow systems. The agents are prescribed about what to do, rather then being more autonomous.
b) The interaction protocol approach provides more autonomy to the agents in the workflow system and the interaction protocol approach has been widely used in the agent-based systems.
c) Interaction protocol approach provides detailed description of interactions between agents as supposed to the process centric approach. It can be observed from figure 3 that the process centric approach hides most of the agent interaction details explicitly in the process model even though it has been implemented programmatically.

It can also be argued that both these approaches can be used together under certain circumstances. The process model shown in figure 3 provides only an overview of the workflow and hides the transition level details from the user. At each transition in the process model, there is some form of information exchange between two agents. For example in the transition *sendRequestToAssessor,* the *ICManager* sends a request to the panel beater to assess the level of damage and send a report. In this case, the process agent that executes this transition can instantiate two agents that adhere to the interaction protocols shown in Figure 2.

Thus, by combining both the ways of modeling, the user is able to understand the workflow in a better way. This provides a richer meaning to the scenario which both these models had failed provide individually, in certain cases.

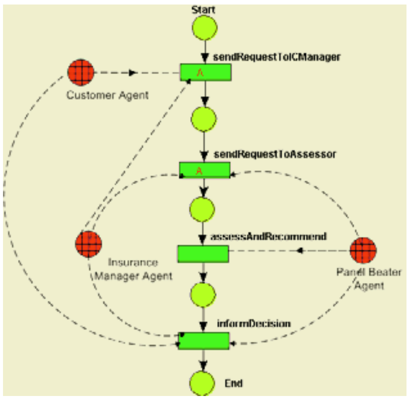

Fig. 3. Process model of the car insurance scenario (process centric approach)

Assume that we now have an e-business workflow scenario such as the English auction, which involves an auctioneer agent, seller agent and the buyer agents. In this case the workflow modeler should choose one of the approaches described in the previous sections. We believe that for this particular scenario the interaction protocol approach would be much more suitable than the process centric approach as the emphasis of the scenario is more on how the interactions between agents take place (bidding, accepting bid etc.) rather than the execution/completion of a particular task. In cases where most of the participants/agents are distributed, the interaction protocol is quite useful, as it would explicitly show the state of each of the participants. In the auction scenario, the state of any of these conversations can be easily traced using an IP model. Though this is possible in the process centric approach, the users will not have an explicit graphical clue to identify the state of the participants.

We would like to mention that when the workflow is oriented more towards processes and completion of various tasks by different roles then the process centric process modelling should be undertaken and for the workflows that have a heavy focus on interactions then the interaction protocol approach should be taken. In cases where the overall process model as well as individual exchanges are important then both mechanism should be used together to provide a richer understanding of the problem at hand. The workflow modelers should understand the different perspectives of modeling mechanisms the system provides and take advantage of them to provide a model that is meaningful to the user of the system.

6 Conclusion

We have presented the two ways of modeling processes, the process centric way and the interaction protocol way. We have also discussed the architectural support for both of these mechanisms in our workflow system. Using a scenario we have described what are the different ways in which a workflow process model can be designed. When the process model has a focus of overall process, the process-centric model is suitable and when the interactions of each participant is important then the modeling should be undertaken based on interaction protocols. When there is a need for understanding both the overall model and the individual interactions, then both models should be used as discussed in section 5. In future we have planned to examine both approaches by using complex models and applying both mechanisms of process modeling and comparing the advantages and the disadvantages.

References

1. R. Scott Cost, Ye Chen, Timothy W. Finin, Yannis Labrou, Yun Peng, Using Colored Petri Nets for Conversation Modeling, Issues in Agent Communication, p.178-192, 2000
2. 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.
3. S. Meilin, Y. Guangxin, X. Yong, and W. Shangguang, 'Workflow Management Systems: A Survey.' in Proceedings of IEEE International Conference on Communication Technology, 1998.
4. 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)
5. W.M.P van der Aalst and K. van Hee, Workflow Management: Models, Methods, and Systems, MIT Press, 2002.
6. Mariusz Nowostawski. JFern – Java based Petri Net framework, 2003.
7. FIPA, FIPA Communicative Act Library - Specification. 2002. http://www.fipa.org/specs/fipa00037
8. 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).
9. 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, pp. 69–76.
10. 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.
11. Savarimuthu, B.T.R and Purvis M.A, 'A Collaborative mulit-agent based workflow system.' In: M. G. Negoita, R. J. Howlett, L. C. Jain (eds.), Knowledge-Based Intelligent Information and Engineering Systems, 8th International Conference, KES2004, Wellington, New Zealand, September 2004, Proceedings, Part II, Springer LNAI 3214, pp. 1187-1193, 2004.