Chapter 6

Creating ontologies for a collaborative, multi-agent based workflow system

Agent based workflow management system has been an active area of research for the known benefits such as collaborative process management in a dynamic and distributed business environment. In this paper we present the architecture of our agent-enhanced and dynamic workflow management system and discuss the creation and usage of ontologies in our system. In particular, we discuss the development of a set of modular ontologies, which correspond to different entities or stakeholders in our system. For example, the terms used in the interaction between various agents that form the building blocks of our system are kept in the *workflow ontology* module. Similarly, the ontology associated with a specific application domain is maintained in a separate module. In addition, the ontology associated with a set of agents that form a particular agent society due to their shared behaviors and concerns are stored separately (such as *seller agents ontology* in an e-commerce application). The relationships between these modules are described in more detail later in this chapter. We also explain how these different types of ontologies are created, managed and used dynamically.

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 (Meilin et.al, 1998, Shepherdson et.al, 1998). 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 (Fleurke et.al, 2003, Savarimuthu et. al 2004a, Savarimuthu et.al 2005b) describe the ad-

vantages of our agent-based framework JBees (Fleurke et.al 2003), such as distribution, flexibility and ability to dynamically incorporate a new process model.

The process models specify the flow of activities in a business process. The meaning associated with various terms and activities specified in these models are defined in ontology. In our system, for each process model, there are a corresponding set of ontology modules. This modular approach with regard to organizing various pieces of knowledge helps the maintenance and dynamic change to the knowledge base. As an example, consider that we are dealing with an e-commerce application selling microwave ovens. If we expand our application, and now include selling computers, we only need to update the application domain ontology to include terms associated with the computer. We do not have to modify the ontology associated with the seller agent (i.e the concepts associated with setting a price, sending a bill, receiving payment, and shipping the product). In this chapter we describe the details associated with various types of ontologies and their relationship with each other.

2 Background

2.1 Coloured Petri Nets (CPNs)

Our system uses Coloured Petri nets (CPN) to model, simulate and execute workflow processes. Petri nets (Murata 1989) are a formalism and have an associated graphical notation for modelling dynamic behaviour of a system. The state of the system is represented by places (denoted by hollow circles) that can contain token (denoted by symbols inside the places). The possible ways that the system can evolve are modelled by defining transitions (denoted by rectangles) that have input and output arcs (denoted by arrows) connected to places. The system dynamics can be enacted (non-deterministically) by determining which transitions are enabled by the presence of tokens in the input places, selecting one and firing it, which results in tokens being removed from its input places and new tokens being generated and placed in the output places.

CPNs (Jensen 1992) are an elaboration of ordinary Petri nets. In a coloured Petri net, each place is associated with a `colour', which is a type (although the theory of CPNs is independent of the actual choice of type system). Places can contain a multiset of tokens of their declared type. When a transition is fired, the matching tokens are removed from the input places, and then multiset expressions annotating the output arcs are evaluated to generate the new tokens to be placed in the output places. If the expression on an output arc evaluates to the empty mul-

tiset then no tokens are placed in the connected place. Figure 1 shows a simple Petri net process model. In our system, CPNs are used to model processes.

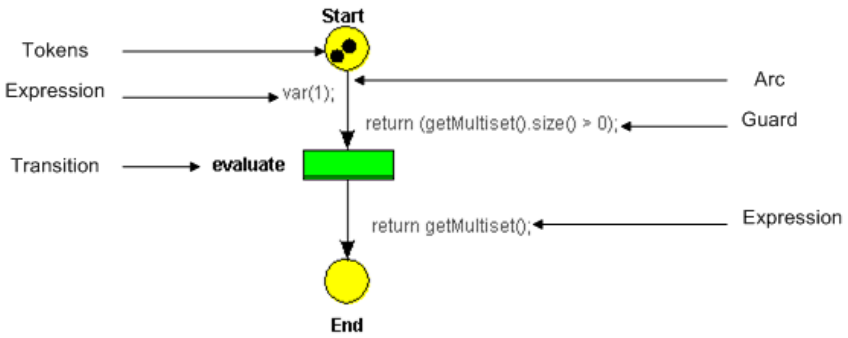

Figure. 1. A simple Petri Net model

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

1. 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).

2. Unlike some event-based process modelling notations, such as dataflow diagrams, Petri nets can model both states and events.

3. 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 (Nowostawski, 2003) - a CPN-simulator and enactment engine to design and execute the models.

2.2 Workflows and multi-agent systems

In the context of WfMSs, agent technology has been used in different ways. 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 (Nissen 2000 and Jennings et.al 2000). An example of this approach is the work by Nissen in designing a set of agents to perform activ-

ities associated with the supply chain process in the area of e-commerce (Nissen 2000). 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 (Stormer 2001 and Wang et.al 2002). 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, where an agent-based WfMS is used in conjunction with specialized agents that provide appropriate application-related services.

In our framework, JBees (Fleurke et.al 2003), we have used the software agents both as part of the WfMS infrastructure as well as entities/resources that perform certain tasks associated with the business domain. Each agent has an interaction protocol model describing how to communicate with the other agents and what actions to perform when a new message (by another agent) is received. In this context the workflow is a set of interaction protocols which are executed towards accomplishing a particular process such as processing an insurance claim or an order entry request to purchase a product.

The agent-based infrastructure facilitates the dynamic incorporation of changed models into the system and thereby is more flexible to the changes that might occur in the environment. The framework, also provides support for inter and intra organizational co-operation in a distributed environment.

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. JBees is based on Opal (Purvis et.al 2002) and uses the CPN execution tool JFern. The processes are modeled using CPNs. A first description of JBees can be found in our previous papers (Fleurke et.al, 2003, Savarimuthu et. al 2004a, Savarimuthu et.al 2005b). Figure 2 shows the architecture of our multi-agent based workflow management system. 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.

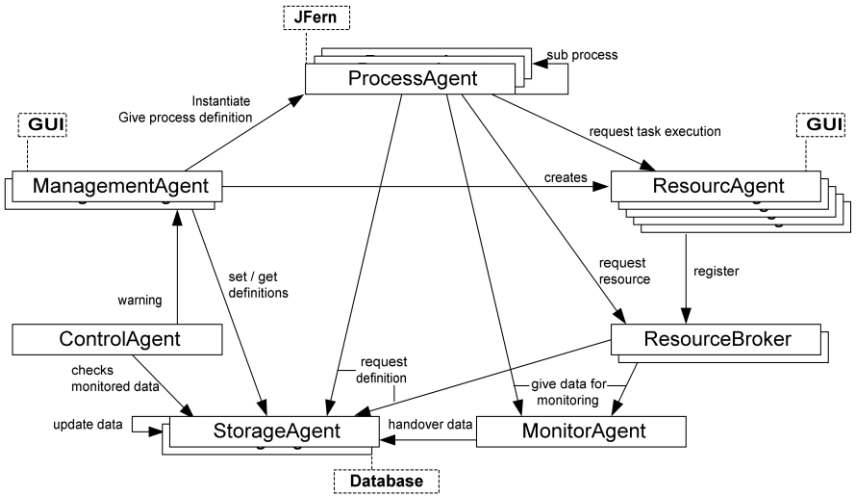

Figure. 2. Architecture of the existing system

Figure 3 shows an order entry process for purchasing a book. 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 the framework works can be found in our previous works.

As noted, the details about some of the more complex task can be elaborated on another sub-process. At this stage the agent responsible for that particular task, could be interacting with other resources/agents in order to accomplish the task. We use the Interaction Protocol approach to model this behaviour (as described in the next section).

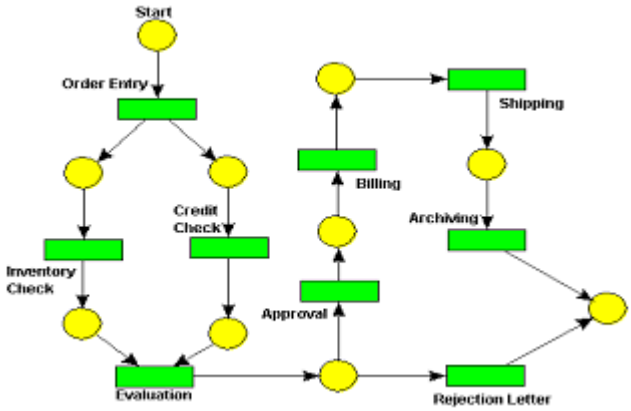

Figure. 3. Order entry process model for purchasing a book online

2.4 The Interaction Protocols (IP)

Interaction Protocols (IP) have been used in the literature for modelling the interactions of an agent with other agents (Scott et.al 2000). Interaction protocols are better intuitive models of how agents will interact than the message based communication between two agents. We use Colored Petri nets to model this interaction. The advantage of using an IP is that an agent is aware of the overall model of the interaction. 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 (Purvis et.al 2002). The transitions that represent the actions can be implemented accordingly depending upon the requirements. The definitions of these actions are represented in the ontology, which is stored in persistently in the distributed databases.

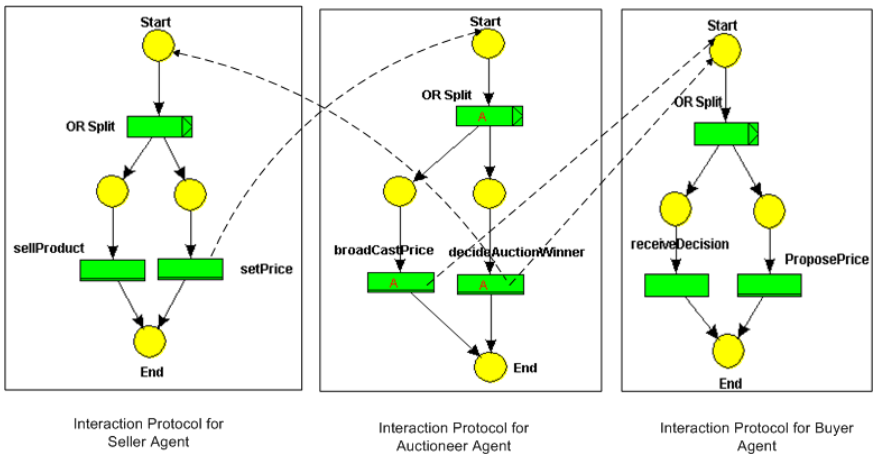

Figure. 4. Interaction protocols of the auction scenario

Figure 4 shows the interaction protocols that each agent executes in an auction scenario involving buyers and sellers. This e-business scenario is expressed using the interaction protocols. The agents involved in the scenario are the seller agent, auctioneer agent and the buyer agent. The seller agent sets the price of the product and sends this information to the auctioneer agent. The auctioneer agent broadcasts the initial starting price to the potential buyer agents with a time out limit. The buyer agents send their bids to the auctioneer agents. The auctioneer agent again broadcasts the reserve price to all the buyer agents till no agent bids. If the final price is greater than the reserve price set by the seller the auctioneer sells the product to the buyer.

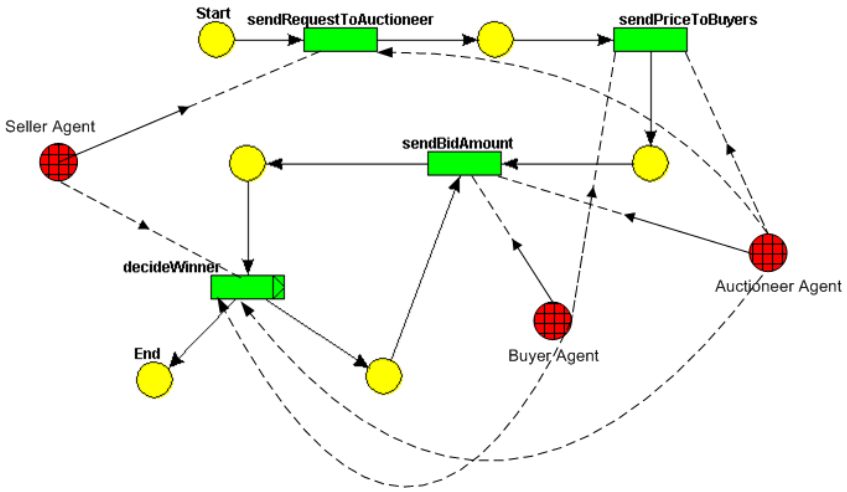

Figure. 5. Overall business process model of the auction scenario

In Figure 4, the modeller has shown specifically the actions taken by each agent and how each agent interacts with the other agents. The auction workflow described in this section uses the interaction protocols. But the same workflow could also be modelled as a process centric model shown in figure 5. It can also be argued that both of these approaches can be used together under certain circumstances. The process model shown in figure 5 provides an overview of the activities included in the workflow. In this model, the actions taken by each agent and how it interacts with other agents may not be as explicit. At each transition in the process model, there is some interactions between the agents involved. For example in the transition *sendBidAmount*, the buyer agent sends a bid amount to the auctioneer agent. In this case, the process agent that executes this transition can instantiate two agents that adhere to the interaction protocols shown in figure 4. Thus, by combining both ways of modeling, the user is able to understand the workflow in a better way (from the overall process view and the individual agent process view). This provides a richer meaning to the scenario that both models had failed to provide individually, in certain cases.

3 Architectural support for ontologies in our framework

Ontologies are used to define the meaning of the terms used in the agent communication. The issues associated with representation and usage of ontologies

have been tackled by several researchers (Cranefield et.al 2003 and Dickinson et.al 2003). There can be many benefits in using ontology including the following:

1) Ontologies are used to build a set of vocabulary for better understanding of the application domain, which can be used consistently throughout the system.

2) Ontologies can facilitate a common understanding of concepts between different agents in the system.

3) By separating the knowledge base from the rest of the process, one can dynamically incorporate a new set of ontologies.

4) When a new agent joins the system, the agent can easily access and process the ontology it has to use.

Figure 6 shows various types of ontologies that are supported and their relationship with each other. These include *workflow ontology*, *agent specific ontology*, *institutional ontologies* and *domain specific ontologies* or *application ontologies.*

In our architecture we have four kinds of ontologies. The architecture is modular which helps organizing various parts of the overall ontology. This modular approach helps easier maintenance and support for the dynamic change in ontologies. The first kind of ontologies are *workflow ontologies*. The ontologies stored in this level is pertinent to the concepts that the workflow framework uses. The next levels of ontologies is *agent specific ontologies*. *Societal/institutional ontologies*, are the third kind of ontologies which store the common concepts that the agents have to understand so that they can be a part of a particular society. The last kind of ontology, the *application ontology*, is specific to the domain under consideration such as Online Purchasing, Apparel Manufacturing, and Software Development etc.

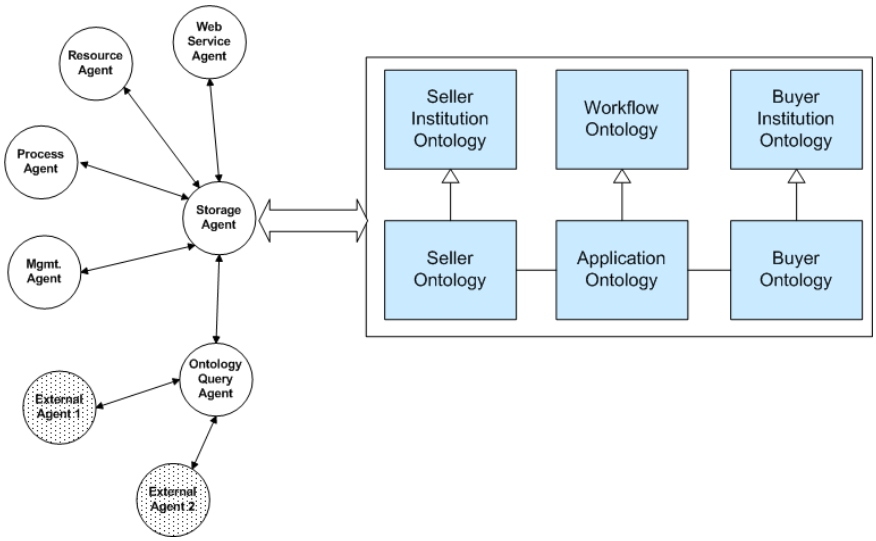

Figure. 6. Ontology architecture of the system

3.1 Workflow ontology

The ontologies for workflow systems are the set of concepts and associated vocabulary and meaning these concepts bring. The workflow management system has concepts that are associated with the building and the operation of the system. The prototype workflow system that we have built has a set of concepts that all the agents, should understand. For example the agents should understand what the following terms such as RESOURCE_IS_AVAILABLE, GET_PROCESS_DEFINITION, GET_TASK_DEFINITION etc. mean in the workflow scenario.

3.2 Agent ontology

The agents involved in an e-commerce scenario in online buying and selling of goods would be buyer agents and seller agents. The buyer sends a request for an item to the seller. In the request the buyer sends the details about the product. The buyer's understanding of the product is stored in ontology and also the terms that the buyer uses in the message. At this level, the ontology is made up of the actions that an agent understands. For example the terms referred to by the buyer agent could be REQUEST_PRODUCT_DETAIL, AGREE_TO_BUY,

MAKE_PAYMENT etc. Ontologies related to the seller agent could be INVENTORY_CHECK, CREDIT_CHECK, BILLING, SHIPPING etc.

3.3 Ontology for societies

The seller agents form societies by adhering to some common rules and concepts such as not selling the defective/expired items. At the society level certain code of conduct may have to be adhered to. For example the members of the society have to follow certain laws when they are engaged in trading certain products in the context of e-commerce such as not to sell alcoholic drink to young customers (as specified in the local laws). Therefore the concept of legal age for purchase of such items will be defined at this level. At this point, we do not have any mechanism for detection and enforcement of legal activities to be carried out by the members of the society.

3.4 Application/Domain specific ontology

The last kind of ontologies are domain specific or application specific ontologies. For an e-commerce scenario that involves buying and selling of products, the concepts that are common to all the lower level entities (such as agents) can be defined. For example, description of the product (make, model etc), the mode of accepted payment when buying a product (credit card/currency used/drafts/cheques) are defined. The concepts associated with the sales tax that has to be collected at the time of the purchase is an issue that is dealt with at this level.

4 Representation of ontologies

In our workflow management system, the workflow modeler creates/updates the relevant set of ontologies when creating the process models.

1) We provide a simple editor to enter the details of the concepts described in the different levels of ontology. The ontology will also provide a mapping of task names to functions. The function will be defined in the same ontology. The user interface that we have provided will obtain the details about concepts (in name-value pairs) as well as the definition of the functions. The ontologies are stored in the XML format (for more details refer to the example provided in section 5).

2) Each higher level of ontology extends the basic level of ontology. For example, the application ontology extends the workflow ontology.

3) For each ontology module, a java class is created from the XML file.

4) The terms and functions used in the process model are resolved by refering to these ontologies through ontology query agent which is aware of the ontology structure. Figure 6 shows the different stages of ontology creation.

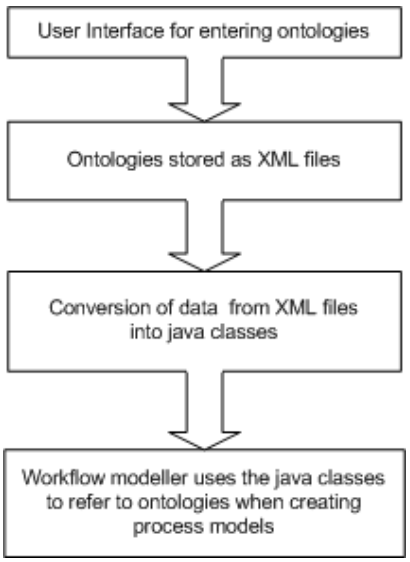

Figure. 7. Flow diagram of the ontology creation process

If a term is not found in the existing ontology structure, it can be added to the appropriate level of the ontology by the authenticated user of the system. When a new agent from an external platform wants to use the ontology, it can do so by contacting the ontology query agent as shown in figure 6. The query agent is familiar with the hierarchical structure associated with particular agent ontology. The query agent would return the name-value pairs of the described ontology to the external agent.

5 Ontology examples

Figure 8 describes a scenario that explains how ontology is used in our frame-work. The process model shown on the left side of the diagram is the process model of a seller agent. When a buyer agent request for a price of the product, the *setPrice* transition is fired. The *setPrice* transition executes the ‘action code’.

Action code is the code that has the instructions about what happens when a transition is fired. In our example the transition invokes the *SellerOntology* to find out the price of the product. The *SellerOntology* checks with the *SellerInstitutionOntology* to verify if the rules laid down by the institution is followed. Each product that is sold should correspond to the description specified in the *BuyerSellerApplicationOntology*. The concepts that are common to the application domain are defined in the *Application Ontology*.

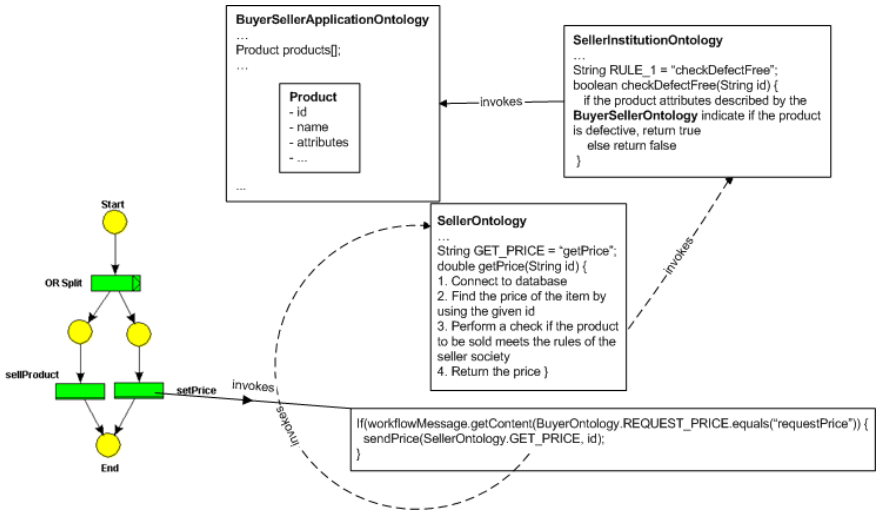

Figure. 8. A simple buyer, seller example to demonstrate the use of ontologies in our system

The *sellProduct* transition produces a bill after considering the details such as the cost of the products purchased and the quantity. The method called *calculateBill* computes the cost associated with this transaction (this is shown in Figure 9). For a more extended version of this example, the details such as tax rate and shipping cost will be considered as well.

To achieve all the above-mentioned details, *sellProduct* transition has to refer to various ontologies such as: Seller Ontology, Buyer Ontology, Seller Institution Ontology, Buyer Institution Ontology apart from Workflow Ontology and Application Ontology. The details that are stored in Seller Ontology include the functions specified in the XML form for *calculateBill, acceptedMethodOfPayment, getPayment, shipProduct, insuranceOptionOffered* and *optionalDiscountRates*.

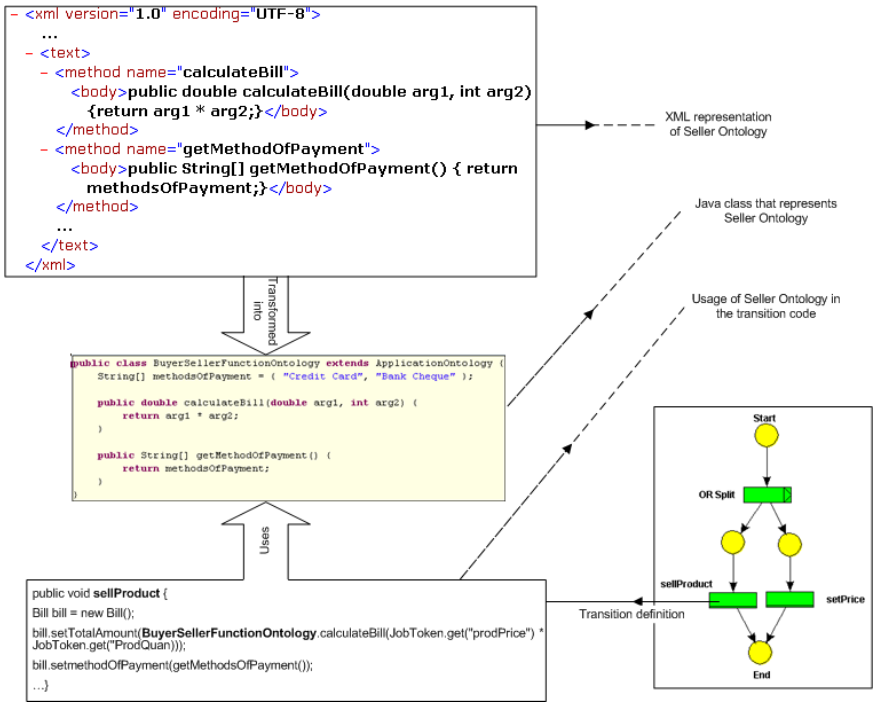

Figure. 9. Creation and usage of ontology for a model that involves the execution of a Java function when a transition is fired

The Buyer Ontology consists of the address of the customer, whether the buyer is a wholesaler or a retailer, the buyers' preferred mode of payment and the bank account information. Seller Institution Ontology will have information regarding various policies such as the refund policy, defective items policy, warranty policy and insurance policy. Buyer Institution ontology will have information regarding the insurance policy governing the purchase of a product. Application ontology governs the common attributes of application such as categories of sale items, product information, shopping cart and the tax rate.

Figure 9 shows the XML representation of the function ontology. This function is invoked during the execution of the process model. The XML ontology is converted to Java class ontology similar to WSDL2java conversion in the context of Web Services domain. As indicated in the diagram, the definition of the Java function is represented as a string in the <body> tag embedded in the XML representation. When a model is designed, the modeler compiles the Java class

that is generated from the XML representation to the Java class format. The workflow modeler will make use of these compiled classes when function ontology has to be called within a transition.

5.1 Handling changes in ontologies

1. One of the advantages of using agents in a workflow system is to dynamically change the process model during runtime. When a process model changes there are chances that new terms can be introduced. If there are new terms defined in the process model, we ask the workflow manager or the user of the workflow system to add these terms or concepts to the ontologies (assuming that these terms are added before the execution of the process model).

2. During the execution of a process model there could be many job instances running. When the ontology changes, the agents executing these job instances will be notified of the change. In the current state of the system, the change is reflected only in those job instances that have started after the changes are committed.

3. In our workflow system we provide mechanisms for the same process model to use different ontologies. An example is the buyer-seller scenario in which special discounts during festive occasions might be offered. In these cases there would be changes to the ontologies with regard to how the selling price of a product is calculated. In our system the agent that executes a process model has an option to choose one of the ontologies out of the several ontologies that are relevant such as *FestiveSeasonSellerOntology* or *RegularSellerOntology*.

4. When a new agent joins a society, it is accepting (can access) the terms and concepts defined in the ontology of that society. When there is any change in the ontology at the societal level (due to a new law) a notification is sent to all the agents registered in the directory facilitator for that particular society.

6 Conclusions

We have explained how we create, store and use ontologies in our framework. We have described the four levels of ontologies in our system. We have explained through the auction scenario example, how these ontologies are used. We have also described mechanisms for handling changes in the ontologies.

We would like to further investigate the incorporation of changes to the ontologies by the agents for currently running instances. Our future work will involve the integration of some of the publicly available domain specific ontologies with the agent based workflow system. We are also planning to design a mechanism to incorporate web service ontologies and hence move towards semantic workflow agents that can use the web services. In case of conflicts in ontologies used by different agents, we would like to explore a mechanism to negotiate with other agents involved in the system before a particular decision is made to update the current ontology in use.

References

Meilin, S, Guangxin, Y, Yong, X, and Shangguang, W (1998), "Workflow Management Systems: A Survey." *Proc. IEEE International Conference on Communication Technology*.

Shepherdson, J.W, Thompson, S.G. and Odgers,B. (1998) "Cross Organisational Workflow Coordinated by Software Agents", *in CEUR Workshop Proceedings No 17. Cross Organisational Workflow Management and Coordination*, San Francisco, USA.

van der Aalst, W.M.P and van Hee, K. (2002), Workflow Management: Models, Methods, and Systems, MIT Press, 2002.

Nowostawski, M.(2003), "JFern– Java based Petri Net framework".

Purvis,M.K, Cranefield,S., Nowostawski,M., and Carter, D (2002), "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.

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.

Fleurke,M., Ehrler, L., and Purvis, M. (2003), "JBees - an adaptive and distributed framework for workflow systems", *Proc. of Workshop on Collaboration Agents: Autonomous Agents for Collaborative Environments (COLA)*, Halifax, Canada, eds, Ali Ghorbani and Stephen Marsh, pp. 69–76.

Savarimuthu, B.T.R., Purvis, M. and Fleurke, M. (2004a). "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.

Savarimuthu, B.T.R and Purvis M.A (2004b), "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

Murata, T. (1989) Petri nets: Properties, analysis and applications. Proceedings of the IEEE, 77(4): 541–580, April 1989.

Jennings, N.R., Faratin, P., Norman, T.J., O'Brien, P. and Odgers, B. (2000), Autonomous Agents for Business Process Management. Int. Journal of Applied Artificial Intelligence, 14(2):145–189, 2000.

Nissen, M.E (2000), "Supply Chain Process and Agent Design for E-Commerce". *In 33rd Hawaii International Conference on System Sciences*.

Stormer, H. (2001), "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.

Wang, M., and Wang, H.(2002), "Intelligent Agent Supported Flexible Workflow Monitoring System". *In Advanced Information Systems Engineering: 14th International Conference, CaiSE*, Toronto, Canada.

Cranefield, S., Pan, J., Purvis, M. (2003), "A UML ontology and derived content language for a travel booking scenario". *OAS*, pp 55-62

Dickinson, I., Wooldridge, M. (2003), "An initial response to the OAS'03 challenge problem". *OAS*, pp 63-70

Scott, R.C., Chen,Y.,Finin,T.W., Labrou,L., Peng,Y. (2000), Using Colored Petri Nets for Conversation Modeling, Issues in Agent Communication, p.178-192

Purvis, M. K., Huang, P., Purvis, M. A., Cranefield, S. J., and Schievink, M. (2002), "Interaction Protocols for a Network of Environmental Problem Solvers", Proceedings of the 2002 iEMSs International Meeting: Integrated Assess-

ment and Decision Support (iEMSs 2002), Volume 3, Andrea E. Rizzoli and Anthony J. Jakeman (eds.), The International Environmental Modelling and Software Society, Lugano, Switzerland, pp 318-323.