Agent-based integration of Web Services with Workflow Management Systems (WfMSs)

Tracking number : 489

Abstract

Rapid changes in the business environment call for more flexible and adaptive workflow systems. Researchers have proposed that Workflow Management Systems (WfMSs) comprising multiple agents can provide these capabilities. We have developed a multi-agent based workflow system, JBees, which supports distributed process models and the adaptability of executing processes. Modern workflow systems should also have the flexibility to integrate available web services as they are updated. In this paper we discuss how our agent-based architecture can be used to bind and access web services in the context of executing a workflow process model. We use an example from the diamond processing industry to show how our agent architecture can be used to integrate web services with WfMSs.

1. Introduction

Workflow management systems (WfMSs) [20, 13] are widely used to manage business processes due to their known benefits such as automation, co-ordination and collaboration between entities. Still, the existing, commercially available workflow management systems do not offer sufficient flexibility for distributed organizations participating in the global market.

Existing systems have rigid, centralised architectures that do not operate across multiple platforms [18]. Improvements can be made by employing a distributed network of autonomous software agents that can adapt to changing circumstances. In particular some of the reasons for wanting an adaptive WfMS are:

- 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 or regulatory environment, new requirements may be imposed which can impact the process definition. Changes in the market may also include the availability of some new technologies and new services such as web services which may require the modification of the process.

The work of Fleurke *et al*.[8, 7] deals with the framework of a distributed network of autonomous software agents that can adapt to the changing circumstances in a workflow management system. The business processes undergo changes over time to accommodate a changing environment such as the availability of web services. Business processes should be able to take advantage of web services that are available in an intranet as well as in the Internet, such as stock monitoring web services. The workflow system that models these business processes should have necessary mechanisms to integrate and use these web services. In this paper we describe the extension to the work done by Fleurke *et al*. The enhanced framework provides mechanisms to create agents that are capable of accessing various web services.

The paper is organized as follows. The next section gives an overview of the background information on coloured Petri nets, software agents,WfMSs and web services. The architecture of the system is described in Section 3. In the fourth section we discuss the underlying mechanism of integrating web services with our workflow system based on multiple agents. We have used a real life example of integrating web services with a diamond processing workflow driven by agents. We present the conclusions and future directions of our work in section five. [1] [2]

2. Background

In this section we explain the background of our work which includes the coloured Petri nets that are used to design the process models, the multi-agent system on which

1 * The primary author is a student.

2 * A demonstration of this system will be available to be shown at the conference.

our workflow system has been built, and some prior approaches in using web services with agents.

2.1. Coloured Petri nets

Petri nets [14] are a formalism and associated graphical notation for modelling dynamic systems. The state of the system is represented by *places* (denoted by hollow circles) that can contain *tokens* (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.

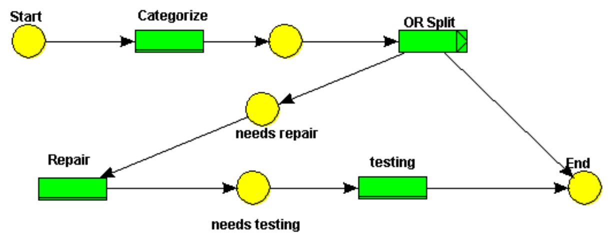


Figure 1: A Coloured Petri net model of fault processing system

Coloured Petri nets (CPNs) [11] 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. Each input arc to a transition is annotated with an expression (possibly containing variables) that represents a multiset of tokens. For a transition to be enabled, it must be possible to match the expression on each input arc to a sub-multiset of the tokens in the associated input place. This may involve binding variables. In addition, a Boolean expression associated with the transition (its *guard*) must evaluate to true, taking into account the variable bindings. 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 multiset then no tokens are placed in the connected place. Our system uses CPN to model, simulate and execute workflow processes. Figure 1 shows a coloured Petri net model of a fault processing system.

2.2. WfMSs and agents

In the context of WfMSs, agent technology has been used in different ways [12]. 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 [10, 15]. 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 [15]. 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 [19, 22]. 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.

In our framework, JBees [8], the WfMS is partitioned among various interacting agents following the interaction protocols. The model associated with a business process is represented using the coloured Petri net formalism and is executed by a specially designed agent. This agent-based environment facilitates the dynamic incorporation of changed models into the system and thereby assists process re-engineering. Advantages of employing agents include the facilitation of inter and intra organizational cooperation and flexibility in choosing processes on-the-fly and controlling process parameters.

2.3. Web Services

Web Services are software components available on the Internet, which provide certain services that may be of general interest, such as weather monitoring services, currency converters, etc. A large fraction of the web services are used within companies protected within their own firewalls. These web services can be accessed for day-to-day business transactions. Examples of these web services include banking services and air ticket booking. The workflow process modeller can integrate web services with the existing workflow sytem. For example, a process model associated with the travel plan of a tourist may depend upon environmental factors, such as the weather conditions. The task associated with finding the weather condition can be provided using a web service.

2.4. Related Work

Some researchers have integrated agent-based workflow systems with web services [4, 5, 21]. However enhancements can be made to improve these approaches.

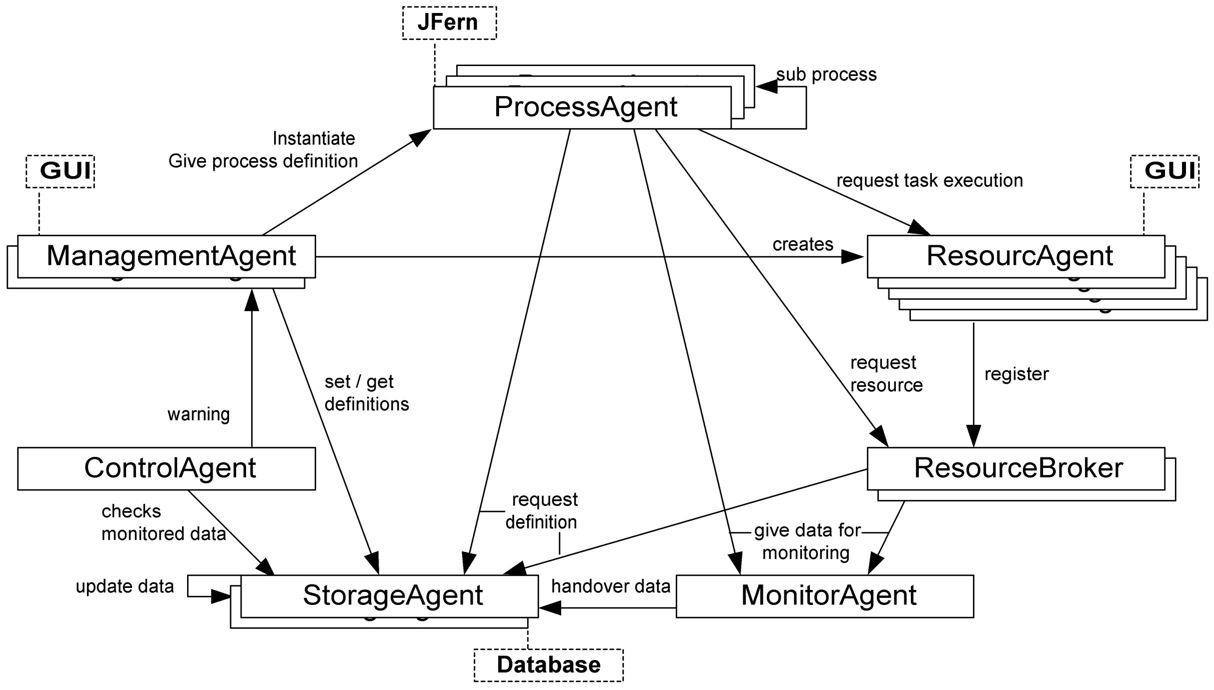


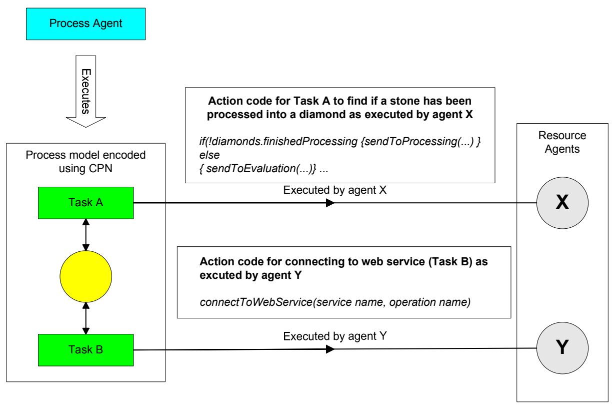
Figure 2: Architecture of the multi-agent based workflow management system

In the research done by Buhler*et al.* [4, 5], BPEL4WS [2] has been used as a process model and this model is converted to a Petri net. The problem with this approach as acknowledged by the authors, is that the demonstration system developed by the researchers so far does not support some of the simple constructs of BPEL4WS. However, in our system the process model is described using a coloured Petri net that can be directly executed. Our system does not require the conversion of a BPEL process into a Petri net process. The conversion mechanism of a BPEL4WS model to a Petri net model has to be validated to ensure the structural and behavioural equivalence associated with the original model.

3. System Architecture

Our system is based on the FIPA [3] compliant agent platform, Opal [17] and uses the CPN-execution tool JFern [16]. Our system consists of seven types of special Opal agents which provide the functionality to control the workflow. Figure 2 shows the architecture of the agent-based workflow system.

The manager agent provides all functionality the workflow manager needs, such as creation and deletion of tasks, roles and process definitions, instantiation of new process instances, and creation of resource agents. The process agent executes a process instance. Each resource in the system has its own resource agent. Every resource in the system gets registered to one of the broker agents that allocate 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 of these criteria to the manager agent. The manager agent provides information to the human manager, which can be used as a feedback mechanism. A detailed description of these agents can be found in the work by *X (name suppressed for blind reviewing purposes)* [24, 23].


Process Agent
Executes
Process model encoded using CPN
Task A
Task B
Executed by agent X
Executed by agent Y
Resource Agents
X
Y
Action code for Task A to find if a stone has been processed into a diamond as executed by agent X
Action code for connecting to web service (Task B) as excuted by agent Y

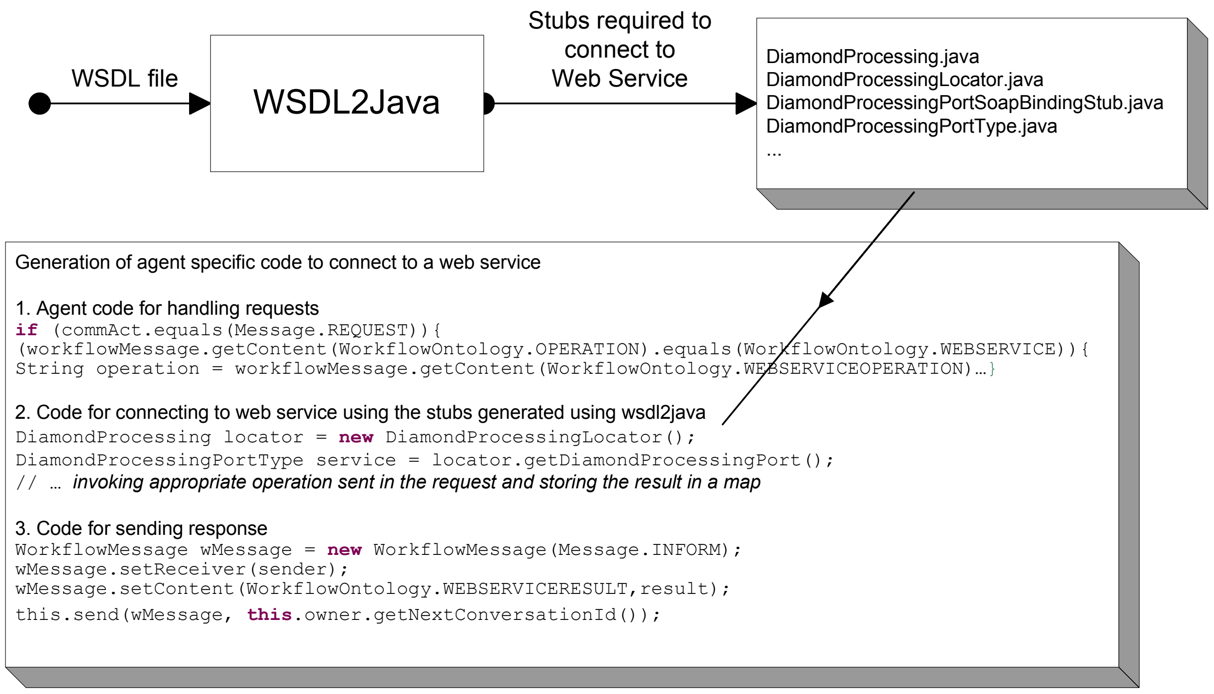
```
if(!diamonds.finishedProcessing {sendToProcessing(...) }
else
{ sendToEvaluation(...)} ...
```

```
connectToWebService(service name, operation name)
```

Figure 3: Task execution mechanism

4. Integrating Web Services

This section describes how we have integrated web services with our workflow management system using agents.


WSDL file
WSDL2Java
Stubs required to connect to Web Service
DiamondProcessing.java
DiamondProcessingLocator.java
DiamondProcessingPortSoapBindingStub.java
DiamondProcessingPortType.java
...
Generation of agent specific code to connect to a web service

1. Agent code for handling requests

```
if (commAct.equals(Message.REQUEST)){
(workflowMessage.getContent(WorkflowOntology.OPERATION).equals(WorkflowOntology.WEBSERVICE)){
String operation = workflowMessage.getContent(WorkflowOntology.WEBSERVICEOPERATION)…}
```

2. Code for connecting to web service using the stubs generated using wsdl2java

```
DiamondProcessing locator = new DiamondProcessingLocator();
DiamondProcessingPortType service = locator.getDiamondProcessingPort();
// … invoking appropriate operation sent in the request and storing the result in a map
```

3. Code for sending response

```
WorkflowMessage wMessage = new WorkflowMessage(Message.INFORM);
wMessage.setReceiver(sender);
wMessage.setContent(WorkflowOntology.WEBSERVICERESULT,result);
this.send(wMessage, this.owner.getNextConversationId());
```

Figure 4: Code generation for a Web Service Agent

4.1. Underlying mechanism

The resource agents in our WfMS are equipped to perform the tasks specified in the transition of the CPN. Each resource subsumes one or more roles. Each role is capable of performing certain tasks.

4.1.1. Types of resource agents. These resource agents may be interfaces to the human agents as well as devices such as scanners and printers. Previous work by X [24] describes how WfMSs support resource agents that interface both humans and computatinal devices. Our objective here is to specialize the resource agent in order to obtain an agent that can perform the task of connecting to any web service.

4.1.2. Execution of tasks. The workflow process models are executed by the process agents. The process agent uses the CPN engine to validate the process model and execute it. A transition in a CPN represents the task that has to be performed. The process agent assigns this task to a resource agent. The resource agent will perform the task and report the result to the process agent. The task to be performed is embedded within the transition of CPN - known as *action code* .

Figure 3 shows an example in which the process agent assigns resource agent X to perform task A. The action code specifies the internal logic of task A. The process agent sends a message to a specialized resource agent Y, which is capable of connecting to a web service to perform the task (task B) specified in the action code. The agent connects to the appropriate operations requested by the process agent on a web service. The agent then sends the result to the process agent.

4.1.3. The Web Service Agent (WSA). In our design we have wrapped the web service as an agent. A specialized agent called Web Service Agent(WSA) is created, and it can be used to query various operations exposed by the Web Service. Thus a resource agent in the WfMS is specialized into a Web Service Agent.

When the process agent executes a process model, a *job token* object is created. This job token consists of a map, that stores various attributes, which include details such as the URI of the WSDL file, and the operation to be invoked on the web service. The workflow manager will create these attributes in the job token.

The attributes encoded in the job token in the form of name-value pairs are:

- Attribute 1: (URIName, URI)
- Attribute 2: (transition name, operation name)

The process agent assigns a task to a resource agent. It passes the job token to the resource agent. The resource agent accesses the URI from the job token and reads the WSDL document and uses WSDL2java from the Axis toolkit [1] to create the necessary stubs. It also generates necessary code for handling requests from other agents to access appropriate operations exposed by the web services and also the code for sending responses to other agents. Figure 4 shows the code that is generated during the creation of a web service agent.

The resource agent sends a message to the process agent about the creation of the Web Service Agent, which is capable of handling requests for operations defined in WSDL. The process agent then assigns the task of connecting and querying the web service, by handing over the job token to the WSA. The web service agent matches the task name from the map and retrieves the operation to be performed. The web service agent connects to the web service and retrieves the result and sends it to the process agent.

In our framework, FIPA ACL [9] is used for agent communication. Figure 5 shows that FIPA messages are exchanged between interacting agents. In the current system the Petri net model provides an abstract view of the process, and the process agent has the built-in intelligence to map the transitions with the agents that are capable of performing the tasks and thereby sending messages to appropriate agents.

4.2. Demonstration of agent-based integration of web service using an example

In this section we describe how a web service has been integrated with the multi-agent based workflow management system. The Petri net model shown in figure 6 illustrates how agents can be used in a diamond processing industry. When the Petri net model shown in figure 6 is executed by the process agent, a token representing a job is created at the *Start* node. The attributes added to the token are given below.

- Attribute 1 : (URIName, http://www.stardiamonds.com/diamonds.wsdl)
- Attribute 2 : (connectAndQueryWebServive, getAllStoneDetails)
- Attribute 3 : (determinePrice, getPrice)

As described in the previous section, the process agent assigns to the resource agent the task of creating a web service agent as inscribed in *createWebServiceAgent* task definition. The resource agent creates a web service agent capable of connecting to the web service as described by the WSDL file given by the URI.

Once the process agent receives a message from the resource agent that a web service agent has been successfully created, the process agent executes the next transition *connectAndQueryWebService* . The operation that is to be invoked is encoded in the job token that is passed along when a transition has been fired successfully. The process agent assigns this task to the web service agent by sending the message to the web service agent with the details about the operation that is to be invoked. The web service agent invokes the webservice, obtains the result, and sends back the result to the process agent as a message. The results are stored in the job token.

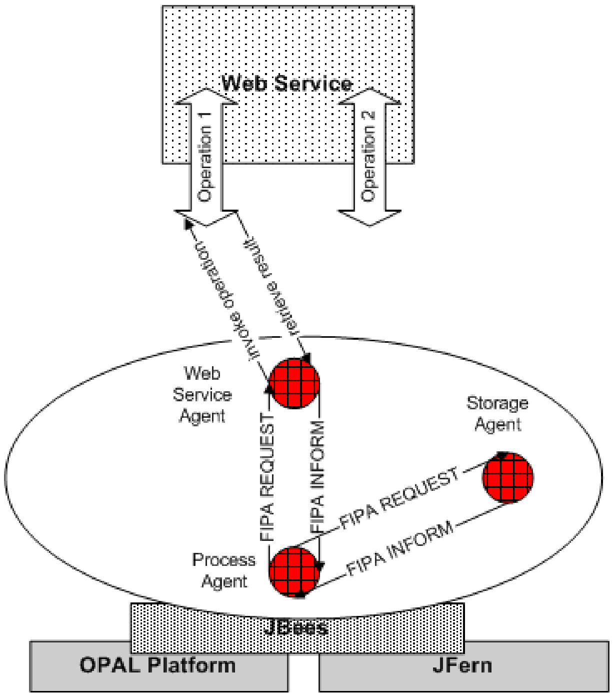
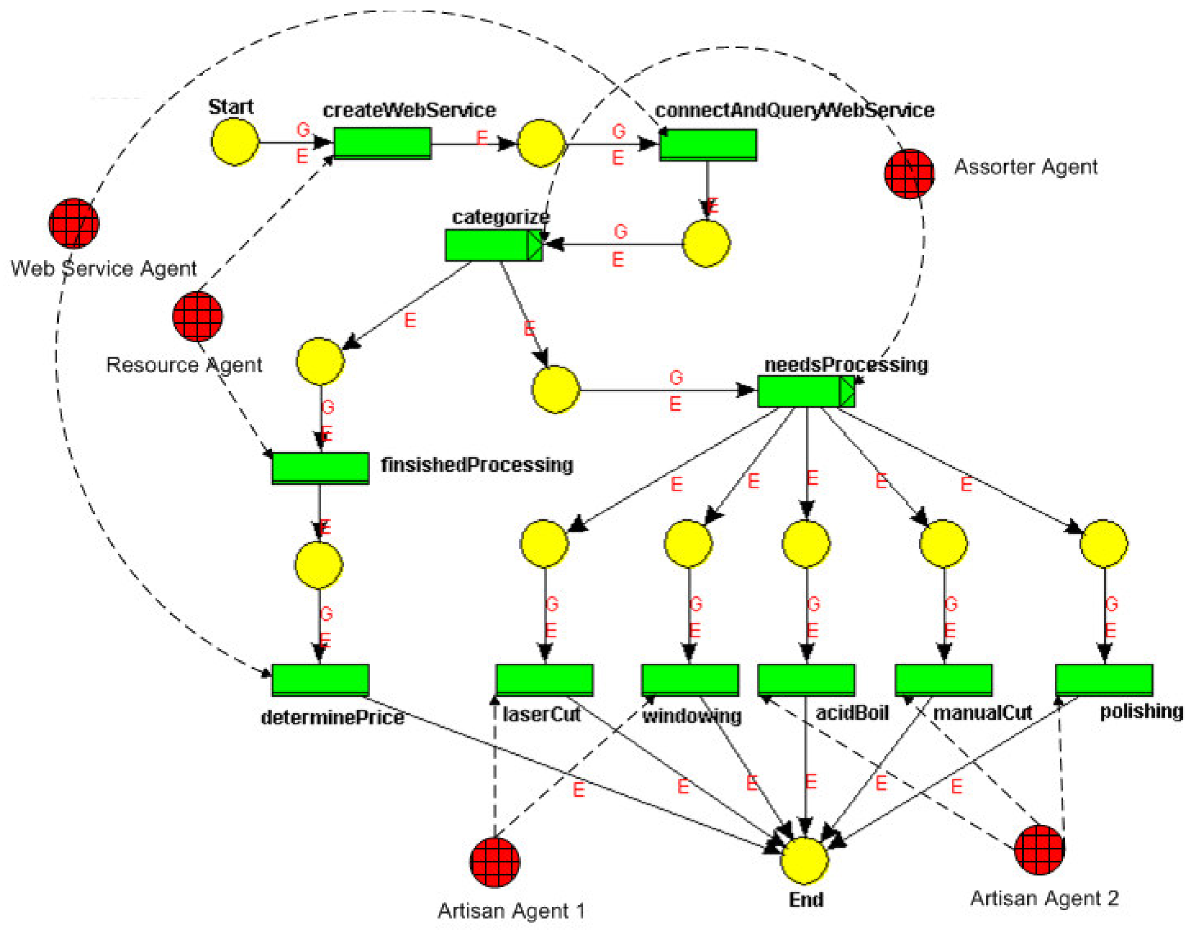


Figure 5: Interactions of a process agent with a web service agent

The result consists of an array comprising of details of the diamonds. For each diamond in the array, a job token is created. A diamond job token would contain details such as *clarity, color, cut, carat weight, lusture, nextProcess, nextArtisan, hasProcessingFinished, price and name-value pairs of the web serivce operations that are to be invoked* . The name represents the transition that is executed and the value represents the operation that is called on the web service. These job tokens are available for the assorter[3] agent who decides if the diamond has been processed. The process agent directs the assorter agent to decide whether each of the diamonds requires further processing. If a diamond requires further processing, the assorter agent determines which process the diamond should undergo and then allocates the appropriate artisan agent to perform that task. If the assorter agent decides that the diamond has been processed completely, it updates the details of the diamond token that can be sent to the evaluator. This is done be setting a value for the boolean attribute *hasProcessingFinished* for that particular token to indicate whether the diamond has been processed.

The web service agent, which acts as a evaluator determines the price of the token depending upon the 4 c's of diamond quality, namely *color, clarity, cut, carat weight*, and assigns a price for that diamond.

3 The job description of an assorter is defined at http://www.occupationalinfo.org/77/770281010.html



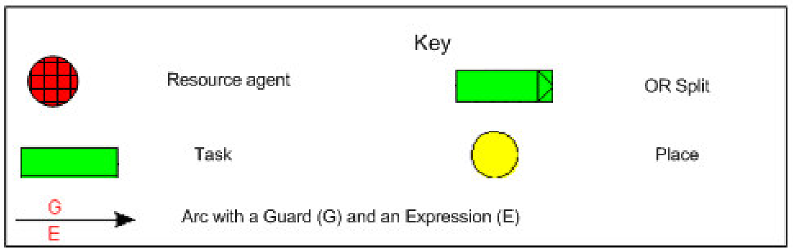


Figure 6: A snapshot of the diamond process model during execution

Each artisan agent is capable of performing one or more tasks. These tasks are *lasterCut*, *windowing*, *acidBoil*, *manualCut* and *polishing*. Each unprocessed diamond is assigned to one of these tasks by the assorter agent. Depending upon the attribute of the job token, these diamonds will be available to a particular artisan agent. The artisan agent performs a particular task on the diamond. After the completion of a task, the artisan agent requests the process agent to add a name-value pair to the attribute list of the job token to indicate the completion of the task. For example, after the completion of a task named laserCut, an attribute binding such as (laserCut, true) is added to the job token.

After a stone has been processed into a diamond that can be sold, it is evaluated and the diamond is made available at the *end* place for further operations such as *dispatchToHeadOffice*. If the processing of the diamond is not complete, the stone details can be obtained from the diamond processing web service. The diamond goes through an iteration of several processes before it is ready for the market. After every task is executed, the storage agent persistently stores all the details of a diamond. Figure 6 shows the pool of resource agents and the process model that is executed. The dotted arrows start from an agent and end in a transition of the CPN. This indicates that the agent is able to perform the task indicated by the transition. Figure 7 shows the sequence diagram for the above mentioned example.

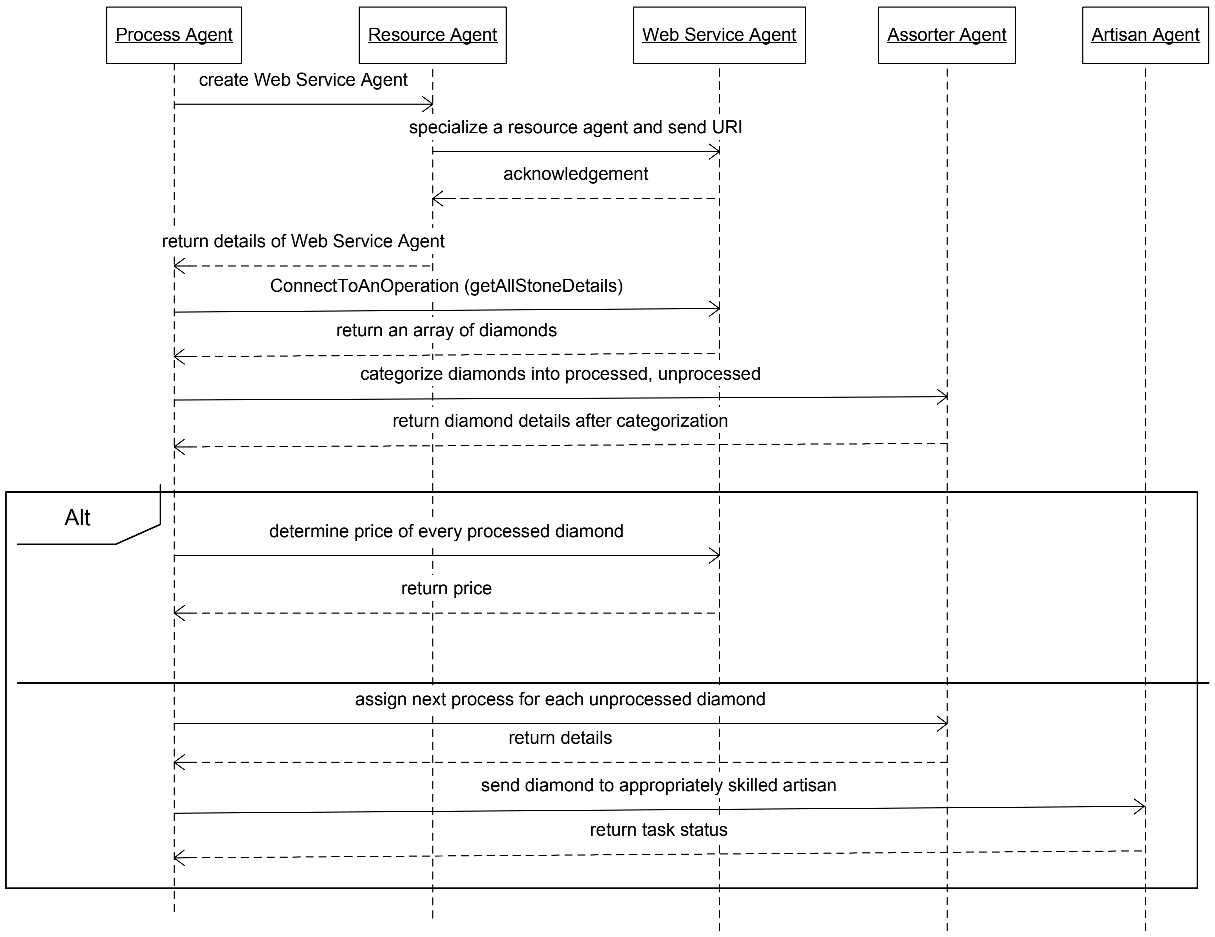

Process Agent
Resource Agent
Web Service Agent
Assorter Agent
Artisan Agent
create Web Service Agent
specialize a resource agent and send URI
acknowledgement
return details of Web Service Agent
ConnectToAnOperation (getAllStoneDetails)
return an array of diamonds
categorize diamonds into processed, unprocessed
return diamond details after categorization
Alt
determine price of every processed diamond
return price
assign next process for each unprocessed diamond
return details
send diamond to appropriately skilled artisan
return task status

Figure 7: Sequence diagram of agent interactions in diamond processing scenario

4.3. Advantages of wrapping a web service as an agent

We believe that modelling a web service as an agent offers the following advantages:

The web service agent speaks FIPA ACL [9]. One of the advantages of modelling the web service as an agent is the ability of the agents in the workflow system to communicate using FIPA ACL [3], messages. This provides a uniform agent-based infrastructural support to the workflow system. Other flexibilities of the web service agent include the storage of the history of successes/failures of web services (which can be used for performance analysis), matchmaking to find web services available on the Internet that offer services specified in WSDL and negotiation with other similar web service agents to perform an operation. The web service agent can be used in the determination of dynamic binding and choosing of protocols. The web service agent is also used to execute the CPN model that models composition of web services since the agent can bind to any web service dynamically. This is achieved by the virtue of the fact that the input to the web service agent is the WSDL description and the required stubs for connecting to the web service are generated during run-time.

Similar to the other agents in the Workflow management system (JBees), the web service agents can be embedded with intelligent decision-making capability, such as choosing the most appropriate operation in a web service or connecting to another web service which offers the same operation, such as contacting a different weather monitoring service to find the temperature of a tourist place of interest. The web service agent can maintain a record of the reputation of Web services and choose the appropriate one dynamically. The web service agents would be able collect and analyze information such as comparing prices of different products across different manufacturers. The agents can also be used for monitoring and controlling, such as constant monitoring for a particular stock increase or decrease or monitoring the level of some reservoir or monitoring temperatures in boilers or reactors. These web service agents can also be specialized into experts, in connection to web services, that belong to particular domains of interest.

5. Conclusion and future work

We have described our flexible, agent based architecture for workflow management systems. The agent based architecture facilitates the easy integration of Web Services with the workflow system. We have demonstrated using an example how a web service can be used in a diamond processing workflow with limited effort. The Web service agent will be able to connect to any Web service dynamically.

In the future we intend to integrate workflow ontologies and domain-specific ontologies to our system and harness the power of the Semantic Web when using the Web Services. We are currently extending our architecture to accomodate a process model that executes composite web services. We are also planning to extend our work to form a society or institution of web service agents that can work collaboratively to achieve a common goal.

6. Acknowledgements

The authors thank Mariusz Nowostawski for his help with the Opal platform.

References

[1] Apache Axis Toolkit . http://ws.apache.org/axis/.

[2] Business Process Execution Language for Web Services (BPEL4WS). http://www.ebpml.org/bpel4ws.htm.

[3] Foundation for Intelligent Physical Agents . http://www.fipa.org.

[4] Paul Buhler and José M. Vidal. Enacting BPEL4WS specified workflows with multiagent systems. In *Proceedings of the Workshop on Web Services and Agent-Based Engineering*, 2004.

[5] Paul Buhler and José M. Vidal. Integrating agent services into BPEL4WS defined workflows. In *Proceedings of the Fourth International Workshop on Web-Oriented Software Technologies*, 2004.

[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] 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.

[8] 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*, pages 69–76, 2003.

[9] Foundation for Intelligent Physical Agents. Fipa communicative act library - specification. http://www.fipa.org/specs/fipa00037/, 2002.

[10] 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.

[11] 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.

[12] 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, 2000.

[13] 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.

[14] Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.

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

[16] Mariusz Nowostawski. JFern - Java-based Petri Net framework . http://sourceforge.net/projects/jfern/, 2003.

[17] 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.

[18] 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.

[19] 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.

[20] W.M.P van der Aalst and K. van Hee. *Workflow Management: Models, Methods, and Systems* . MIT Press, 2002.

[21] José M. Vidal, Paul Buhler, and Christian Stahl. Multiagent systems with workflows. *IEEE Internet Computing*, 8(1):76–82, January/February 2004.

[22] 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.

[23] X and Y. A collaborative multi-agent based workflow system. In *Knowledge-Based Intelligent Information and Engineering Systems (KES)*, pages 1187–1193. Springer, 2004.

[24] X, Y, and Z. Monitoring and controlling of a workflow management system. In *In Proc. Australasian Workshop on Data Mining and Web Intelligence (DMWI2004)*, pages 127–132, 2004.