Integrating Workflow into Agent-Based Distributed Data Mining Systems

Chayapol Moemeng, Xinhua Zhu, and Longbing Cao

Quantum Computing and Intelligent Systems,
Faulty of Engineering and Information Technology,
University of Technology, Sydney
{mchayapol,xhzhu,lbcao}@it.uts.edu.au

Abstract. Agent-based workflow has been proven its potential in overcoming issues in traditional workflow-based systems, such as decentralization, organizational issues, etc. The existing data mining tools provide workflow metaphor for data mining process visualization, audition and monitoring; these are particularly useful for distributed environments. In agent-based distributed data mining (ADDM), agents are an integral part of the system and can seamlessly incorporate with workflows. We describe a mechanism to use workflow in descriptive and executable styles to incorporate between workflow generators and executors. This paper shows that agent-based workflows can improve ADDM interoperability and flexibility, and also demonstrates the concepts and implementation with a supporting the argument, a multi-agent architecture and an agent-based workflow model are demonstrated.

1 Introduction

Workflow is an automation of a business process during which document, information or tasks are passed from one participant to another for action, according to a set of procedural rules [1]. The workflow metaphor is widely accepted and used in various process modeling tasks, such as service, production, security, data mining, etc. With regard to knowledge discovery in databases (data mining, for short), most existing data mining tools, such as SAS Enterprise Miner , PASW , Weka[2], RapidMiner[3], KNIME[4], provide workflow metaphor for process visualization, audition, and monitoring. Process visualization is incredibly useful particularly in distributed environments in which complex process configuration can be quickly and correctly interpreted. In agent-based distributed data mining (ADDM), coordination between distributed nodes is a complex task, the use of workflow metaphor is beneficial to both users and systems mutually. Besides improving user friendliness, the system can use the workflow in coordinating the process. In this work, we integrate the use of workflow with the process coordination in ADDM.

In fact, a workflow is only a document describing the process execution. In order to cope with dynamic changes in resource levels and task availability, the workflow engine is the one who actually deals with those difficulties. Major difficulties relate to (i) dynamic changes of the system environment as well as (ii) workflow execution exceptions [5]. Firstly, system resources, such as CPU, disk storage, security access, etc., may

L. Cao et al. (Eds.): ADMI 2010, LNCS 5980, pp. 4–15, 2010.
© Springer-Verlag Berlin Heidelberg 2010

change over time. The plan to execute a workflow must be prepared prior to the execution, because execution plan depends on the current situation hence derived differently over time. Secondly, while a workflow is being enacted, failures during an execution can possibly stall the entire process. Thus, workflow engine must handle various kinds of exceptions and recover so that the execution can be carried on completely.

Attempts to overcome workflow's difficulties with agents have shown very promising results [6–11] by integrating agents into workflows, as known as agent-based workflow (AWF). AWF is a style of workflow management that allows workflow to be fully provisioned by agents in which agents provide solutions to the previously addressed difficulties of traditional workflow management. Regarding dynamic changes of resources, agents autonomously perceive the current situation of the environment and develop proper settings appropriately to the changes of the environment, for example, selecting a network communication channel from current available choices at the run time. Concerning exception handling, a generic approach is to provide a control and monitoring mechanism for the workflow management. However, in a distributed environment, the centralized control and monitoring mechanism fails to take immediate action when the network communication is not available, while some exceptions can be solved by simply re-running the workflow activity again [10]. Agents' pro-activeness is suitable when trivial decision can be made immediately without consulting the central controller. As a result, decentralized workflows have been investigated further and found potential extensibility when integrate with multi-agent system. The critical features of AWF systems are autonomous collaboration and communication [7]. Agents must cooperate, communicate and negotiate with other agents for coordinating the workflow and executing sub tasks of the workflow on various locations. In this work, we argue that AWF can improve ADDM's flexibility and interoperability. Consider an ADDM system, agents are already an integral part and are assigned to various responsibilities, e.g., performing data mining task on remote sites, etc. Therefore, adding an extra responsibility or agents to the existing framework does not cost as much as re-engineering the whole system structure. In conclusion, the integration of workflow into ADDM can enhance ADDM in terms of the following aspects.

First, the flexibility of the system is determined by how well the system can adapt to new tasks. Variations in data mining process can complicate the system, thus workflows are used to represent these variations in a form that can be interpreted by agents. Agents' planning evaluates a descriptive workflow into an executable one which describes specific details, e.g., resource location, related agent containers, etc.

Second, the interoperability concerns how agents work with each other. Given a data mining workflow, an original workflow can be evaluated into different versions depending on the current system situation, e.g., system resources, available agents, etc. Therefore, the best execution plan can be pre-determined prior to the actual execution for the best interoperability. The rest of paper is organized as follows: section 2 shows a review of related work and discusses their limitations, section 3 describes how AWF integrates with ADDM, section 4 shows the experiment system for the integration, section 5 describes the implementation and an example, section 6 is the evaluation, and finally the paper concludes in section 7.

2 Related Work

Existing data mining tools, e.g. SAS Enterprise Miner, PASW, Weka [2], RapidMiner [3], KNIME [4], use workflow to visualize data mining process. SAS, PASW, RapidMiner and KNIME also provide enterprise version of their products distribute its work over multiple processors and to perform simultaneous calculations on multiple models in the client. Their client/server architecture includes workflows that also involve distributed data analysis processes. RapidMiner Enterprise Analytics Server and KNIME Enterprise Server aims to provide data analysis service thru the Internet which users can access the service via web browsers. All computations occur at the enterprise server. Another KNIME enterprise product, Cluster Execution, aims to increase computing power using computer clusters. These systems assume that data must be visible to the entire system; therefore one node cannot hide its data from another.

With regards to AWF, many researchers have been exploring the use of agents to improve process integration, interoperability, reusability and adaptability in distributed environment. [12] analyses benefits of integrating workflow and agent technology for business process management. [10] presents the monitoring and controlling aspects of distributed agent-based workflow systems. [13] develops a multi-agent-based workflow system that supports distributed process models using web services. [8] presents an agent-based approach for coordinating product design workflows that supports distributed dynamic process management. [7] argues that in order for agents to communicate effectively and enhance interoperability, they need to have mutually understandable and standard semantic constructs, and they propose an RDF-based schema to define communication semantics among agents. Unfortunately, no research has applied to the DDM scenario.

Agents have proven its excellence in facilitating particular needs in DDM. A few classic systems, discussed in [14], have demonstrated the use of agents in distributed learning environment, e.g. JAM, PADMA, BODHI, etc. Agents offer desirable properties suitable for complex systems, for example, agent sociability enables DDM to choose appropriate platforms to execute the actions; that is to allow agents to perform locally on the data site without exposing the data. Unfortunately, none of these systems support workflows. Despite both AWF and ADDM researches have been prototyped; still no research presented the integration of both technologies.

3 AWF for ADDM

Data mining process is often customized and varied by the requirement of each particular data mining task. The process can be described in a modeling language, such as PMML [4], as a mean to store and retrieve for later use with other compatible tasks. The main idea of this work is to merge data mining model and workflow into one document that can be used by ADDM.

Initially, the workflow is presented in a descriptive style document which contains relative information about data mining process, such as resource name, relative path of data, etc. The descriptive workflow is then evaluated to an executable style document which contains specific information, such as physical resource location, absolute path

of data, specific agent container. In this way, the responsible unit that creates descriptive workflows (i.e. external entity) does not need to comprehend the knowledge about the system; instead the execution unit which resides internally in the system is responsible for the workflow evaluation. The workflow evaluation is carried out prior to the execution which results in different versions depending on the current situation of the system. Thus the system configuration is protected from external entity.

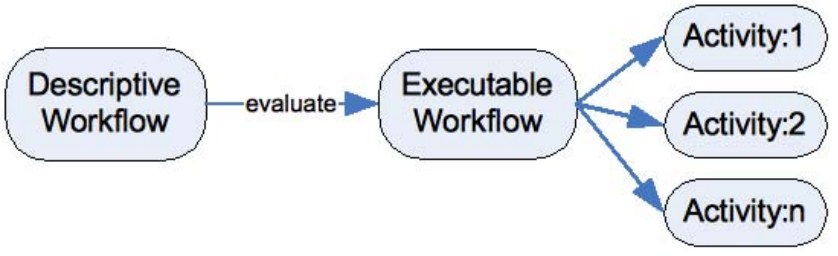

Fig. 1. Workflow transformation

3.1 i-Analyst: An ADDM System

For experimenting purpose, we have created an ADDM system, named i-Analyst. The system aims to provide data mining algorithm testing environment. Each test case is represented as a data mining model (DMM) which describes data mining process as a workflow. Workflows in i-Analyst composes of series of activities, such as data acquisition, transformation, mining, evaluation, and presentation. The infrastructure allows testers to construct their models to test arbitrary algorithms to operate on available data. Both algorithms and data can be uploaded into the system and become available for sharing.

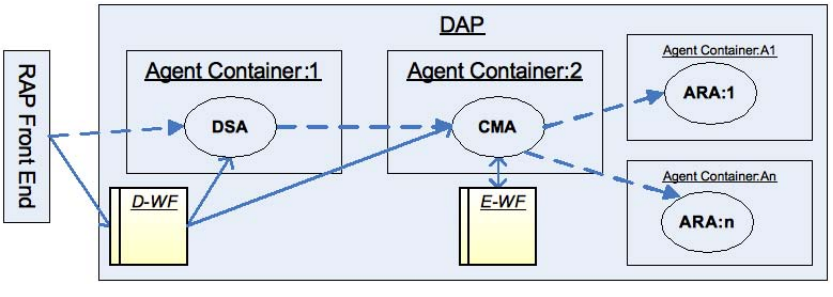

Fig. 2. The i-Analyst architecture

From the technical aspect, i-Analyst is a multi-agent system (MAS) that distributes sub-tasks to run on distributed platforms. The system composes of two core parts: Rich Ajax Platform (RAP) and Distributed Agent Platform (DAP) as shown in figure 2. Java

and its related technologies, e.g. Eclipse for RAP, Jade for agent SDK, are key tools for the system development.

RAP is a web-based front-end system that interacts with the user, while DAP is the automatic background process containing multiple agent containers. The main resources of the system are data mining algorithms, data sets, and reports. The three combines as a running instance, that is, mining data sets using a particular algorithm and producing a report. An instance (in this context, data mining model) is a workflow of data mining tasks. DAP consists of three main agents:

- DAP Service Agent (DSA) is the entry point that interacts with RAP. DSA chooses an appropriate Case Mediator Agent to carry out an instance execution.
- Case Mediator Agent (CMA) evaluates the descriptive workflow to a detail execution plan (executable workflow), controls and monitors workflow execution, and interacts with Activity Runner Agents to run activities.
- Activity Runner Agent (ARA) runs an activity assigned by a CMA.

4 Data Mining Model

A generic workflow composes of activities and transitions [15]. Each activity receives inputs and produces outputs. Outputs from the prior task can be used as inputs for the subsequent task only if the structures of inputs and outputs are compatible. Transition is triggered by events, usually task completion or conditions.

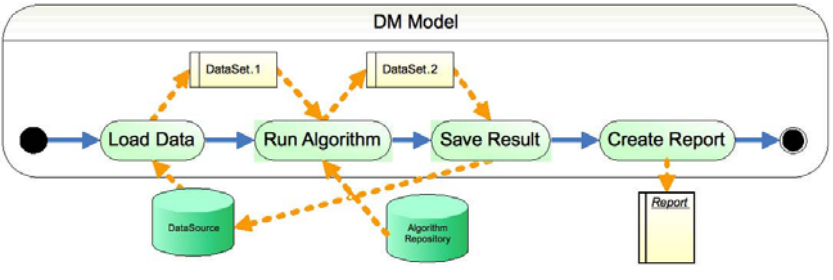

Fig. 3. A simple data mining model

Data mining model (DMM) in i-Analyst is represented as a workflow document which is based on a specific XML schema. DMMs are carried out by DAP. Figure 3 shows a simple data mining model defines a series of activities: LoadData, RunAlgorithm, SaveResult, and CreateReport. Each activity may produce any arbitrary data sets and store them in the model memory for other activities to consume. Sharing inputs and outputs does not occur in a single memory unit, due to distributed environment of the system. Inputs and outputs are determined to be transferred from one agent to another

agent as needed. Since each activity is run by an ARA; ARA stays alive until there is no dependency to it.

5 Descriptive Workflow Evaluation

The schema of DMM is based on an XML Schema Definition (XSD); figure 4 shows a simplified schema. Model is the data mining model itself which contains activities and data sets. Activity is an abstract class which defines generic structure of an activity. Each activity receives input and output dataset; while parameter specification is defined in the actual DMM contents. LoadData, SaveResult, RunAlgorithm, and CreateReport are sub classes extending from Activity. Each concrete activity defines its required parameters that are configured specifically for each model instance (describe in the later section). Figure 5 shows a sample content of LoadData in descriptive format. A system variable, i.e., ${dataset_dir} is to be replaced with actual values of each agent environment. In this example, ARA loads a data set from a CSV file located at ${dataset_dir}/dpriceData.csv into a dataset dpriceData.1.

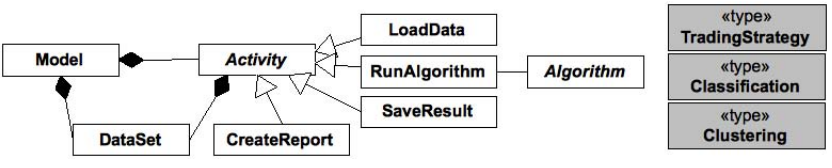

Fig. 4. simplified DMM schema

```
<activity id="1" type="LoadData" name="LD1" nextActivityIds="2">
   <inputDataSources>
      <dataSource id="1111" name="dpriceData.1" type="CSV"
        path="${dataset_dir}/dpriceData.csv">
          <fields>...</fields>
       </dataSource>
   </inputDataSources>
</activity>
```

A sample content of LoadData in descriptive format.

Another example, i-Analyst allows users to upload new algorithms to the system; the algorithms are distinguished by namespaces. An algorithm may contain several related classes, these classes are packed into a file, in this case Java Archive (JAR) file. A descriptive algorithm name only defines fully qualified name of the algorithm including namespaces to be used in the data mining model, e.g., ianalyst.algorithm.Spam. The evaluation process for the algorithm name will add additional information, that are (i) the file path where the algorithm JAR file is located and (ii) the agent name which

is assigned to run this algorithm depending on the algorithm requirements, such as language and libraries. For example,

```
<algorithm name="ianalyst.algorithm.Spam"/>
```

is evaluated to

```
<algorithm name="ianalyst.algorithm.Spam"
 jar="/contents/algorithm/uploads/jar001.jar aid="ara001">
```

Similar to the algorithms, a descriptive data source only denotes the name with fully qualified name, e.g., stock/marketA/dpriceData2009. Other additional configurations can also be supplied, for instance, localAccessOnly indicates whether the activity should be executed on the remote platform where the database is accessible, hence the same network. For example,

```
<dataset name="stock/marketA/dpriceData2009" localAccessOnly="true"/>
```

is evaluated to

```
<dataset name="stock/marketA/dpriceData2009"
 uri="jdbc:mysql:http://analyst.it.uts.edu.au:3306/ianalyst"
 aid="ara001">
   <fields>...</fields>
</dataset>
```

In case of view data sets, the data set that is generated from data processing instructions, e.g. SQL statements, the executable form will be as following

```
<dataset name="stock/marketA/dpriceData2009"
 uri="jdbc:mysql:http://analyst.it.uts.edu.au:3306/ianalyst"
 aid="ara001" dpi="true">
  <dpiSequence>
    <dpi type="sql" key="proc01" value="SELECT * FROM stock WHERE date
       BETWEEN '2009-01-01' AND '2009-12-31'"/>
  </dpiSequence>
  <fields>...</fields>
</dataset>
```

6 Implementation

The agent collaboration model is another critical issue which significantly determines the entire system performance. In this section, we describe how agents work in the system.

6.1 Agent Collaboration Model

Figure 6 shows the sequence diagram for agent message passing. The process starts from the DSA receives a request to run a DMM instance from RAP. It then searches for

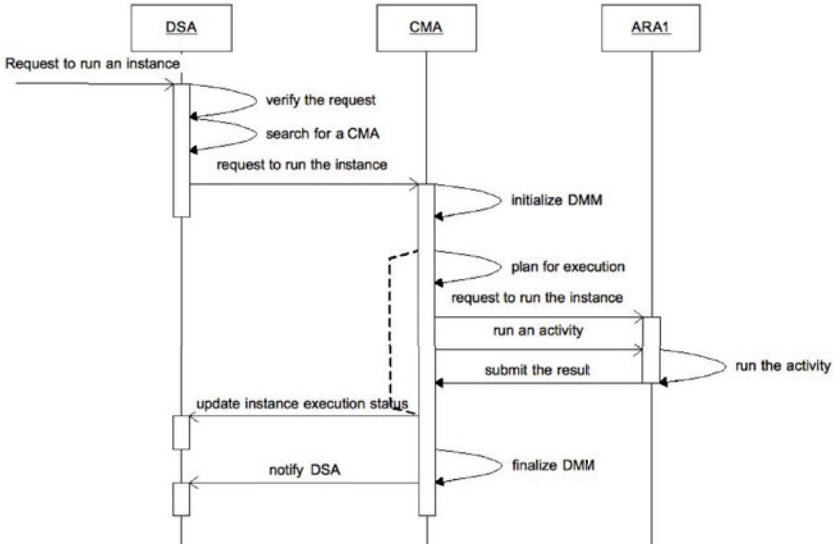

Fig. 5. Sequence diagram for agent message passing

a CMA to run the instance. The CMA receives the forwarded instance running message and retrieves the DDM instance from the system database. The CMA evaluates the DDM instance, in this context, the descriptive workflow, to the executable workflow. The CMA then spawns respective ARAs to run activities. An ARA runs an activity received from the CMA. It requests for required inputs from the CMA, in which the CMA may direct the ARA to an appropriate source if the CMA does not have an access to the data. The ARA completes the activity and produces outputs. A security constraint is applied here; the ARA determines whether to send the outputs back to the CMA or possesses it for future request. The CMA receives completion notification from ARA and forwards the notification to the DSA.

6.2 Example

In this section, we demonstrate how the concept is implemented. From figure 7, system variables, e.g., ${project_classpath}, ${dataset_dir}, ${result_dir}, ${report_dir}, are to be replaced with local values of each agent environment. In this example, we load a data from a CSV file located at ${dataset_dir}/dpriceData.csv into a dataset dpriceData.1. The dataset is visible throughout the model and of course shared with other activities. RA1 requires dpriveData.1 hence CMA sends the dataset to the RA1 ARA. Result of applying MovingAverage algorithm is the transferred into the dataset result. The last activity is to save the dataset into a file. In this example, the model exports the dataset to ${result_dir}/resultData.csv.

```
<instance id="100679" name="Instance1">
  <dataSets>
    <dataSet id="12345" name="dpriceData.1">
      <fields>
        <field name="trading_date" type="xs:string" />
        <field name="sum_daily_price" type="xs:decimal" />
      </fields>
    </dataSet>
    <dataSet id="54321" name="result"><fields>...</fields></dataSet>
  </dataSets>
  <activities first_activity_id="1">
    <activity id="1" type="LoadData" name="LD1" nextActivityIds="2">
      <inputDataSources>
        <dataSource id="1111" name="dpriceData.1" type="CSV"
          path="${dataset_dir}/dpriceData.csv">
          <fields>...</fields>
        </dataSource>
      </inputDataSources>
      <outputDataSets><dataSet id="12345" /></outputDataSets>
    </activity>
    <activity id="2" type="RunAlgorithm" name="RA1"
      previousActivityIds="1" nextActivityIds="3">
      <algorithm id="2222" name="MovingAverage"
        path="${project_classpath}"
        className="ianalyst.algorithm.impl.MovingAverage">
        <parameters>
          <inputParam key="m" value="5" type="xs:int" />
            ...
        </parameters>
      </algorithm>
      <inputDataSets>
        <dataSet id="12345">
          <fieldMapping>
            <field localName="date" orgName="trading_date" />
            <field localName="price" orgName="sum_daily_price" />
          </fieldMapping>
        </dataSet>
      </inputDataSets>
      <outputDataSets><dataSet id="54321" /></outputDataSets>
    </activity>
    <activity id="3" type="SaveData" name="SD1"
      previousActivityIds="2" nextActivityIds="4">
      <inputDataSets><dataSet id="54321" /></inputDataSets>
      <outputDataSources>
        <dataSource id="3331" type="CSV"
```

```
            path="${result_dir}/resultData.csv">
            <fields>...</fields>
          </dataSource>
        </outputDataSources>
      </activity>
      <activity id="4" name="CreateReport1"
        previousActivityIds="333" type="CreateReport">
        <dataFile path="${result_dir}/tt"/>
        <reportTemplate id="4444" name="CR1"
          path="${reporttemplate_dir}/ReportTemplate/RT01.zip">
          <fieldMapping>
            <fields>...</fields>
          </fieldMapping>
        </reportTemplate>
        <report path="${report_dir}/Report/yy" type="PDF"/>
      </activity>
    </activities>
  </instance>
```

Fig. 7. Content of a DMM.

The descriptive DMM content in figure 7 reflects the figure 5. The execution process takes place as follow. The chosen CMA is promoted to carry out the workflow. The DMM is evaluated to an executable workflow as described in section 4.3. The executable workflow which has four activities is and monitored by the CMA. The CMA requests target platforms to spawn respective ARA. The CMA sends activity1 to the respective ARA1, in which the ARA must be on the same site where the data is available. ARA1 loads data from into the memory, maintains in its memory, and notifies the CMA of the completion. The CMA notifies ARA2 to run activity2. The CMA sends the algorithm content to ARA2. ARA2 acquires for input dataset which was produced by activity1. CMA notifies ARA1 to send the data to ARA2. In the planning stage, ARA1 and ARA2 are placed in the same or neighbor network to minimize the communication cost. ARA2 applies the algorithm on the data, then produces a result set, and notifies CMA for the completion. CMA notifies ARA3 to starts activity3. ARA3 fetches required data and saves to its local directory which is also visible to ARA4. After the completion of ARA3, ARA4 receives notification from the CMA and starts activity4. ARA4 merges the output data and a report temple into a report document, and places on the designated directory. ARA4 notifies the completion to the CMA. The CMA receives all completion confirmations and finally notifies the DSA for the workflow completion.

7 Evaluation

The integration method augments the existing ADDM architecture by introducing additional responsibility to the agents to perform according to the assigned activity. The data mining models are improved to be used as workflows, so the system only maintains only one document, the descriptive one. The workflow is used to help planning of agents. With descriptively provided workflow, the responsible agent (CMA) can insert more

details to the workflow for the execution. The benefit is that the system does not need to disclose internal information for the descriptive workflow producer, e.g., third-party front-end system, therefore data and system configuration are hidden. Model evaluation determines the resource utilization. Planning may concerns the cost of communication and computation time. Planning may result in a plan to execute all activities locally relatively to the CMA if the size of data is too excessive to transfer across the networks. A group of consecutive activities may be executed on the same containers as to minimize data transmission cost. In terms of robustness, exception handling is achieved through agent negotiation. Agents can be programmed to handle pre-deterministic exception internally, before acquiring for others assistance. The system benefits from less interruption from exceptions that can be handled. Non-deterministic exceptions are forwarded to the CMA and the CMA decides to start the activity else where.

8 Conclusions

We present an integration of agent-based workflows with agent-based distributed data mining. The main idea is to blend the two technologies together by treating the data mining models as workflows. Since agents are already an integral part of the system, augmenting the agents responsibility to enact the workflow does not require revising the entire system structure. We have demonstrated the use of workflow in our existing system which is used for the ongoing research focusing on supporting more complex data mining tasks and how to reflex the model schema as agent ontology as to improve the system implementation when the two concepts are closely mapped.

References

1. Specification, W.M.C.: Workflow Management Coalition Terminology & Glossary (Document No. WFMC-TC-1011). Workflow Management Coalition Specification (1999)
2. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: An update. SIGKDD Explorations 11(1), 10–18 (2009)
3. Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., Euler, T.: Yale: rapid prototyping for complex data mining tasks. In: KDD '06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 935–940. ACM, New York (2006)
4. Berthold, M.R., Cebron, N., Dill, F., Gabriel, T.R., Kötter, T., Meinl, T., Ohl, P., Sieb, C., Thiel, K., Wiswedel, B.: KNIME: The Konstanz Information Miner. In: Studies in Classification, Data Analysis, and Knowledge Organization (GfKL 2007). Springer, Heidelberg (2007)
5. Shepherdson, J.W., Thompson, S.G., Odgers, B.R.: Decentralised workflows and software agents. BT Technology Journal 17(4), 65–71 (1999)
6. Cai, T., Gloor, P.A., Nog, S.: Dartflow: A workflow management system on the web using transportable agents (1997)
7. Huang, C.J., Trappey, A.J., Yao, Y.H.: Developing an agent-based workflow management system for collaborative product design. Industrial Management and Data Systems 106(5), 680 (2006)
8. Judge, D.W., Odgers, B.R., Shepherdson, J.W., Cui, Z.: Agent-enhanced workflow. BT Technology Journal 16(3), 79–85 (1998)

9. Odgers, B.R., Shepherdson, J.W., Thompson, S.G.: Distributed workflow co-ordination by proactive software agents. In: Intelligent Workflow and Process Management. The New Frontier for AI in Business IJCAI-99 Workshop (1999)
10. Savarimuthu, B.T.R., Purvis, M., Fleurke, M.: Monitoring and controlling of a multi-agent based workflow system. In: ACSW Frontiers '04: Proceedings of the Second Workshop on Australasian Information Security, Data Mining and Web Intelligence, and Software Internationalisation, pp. 127–132. Australian Computer Society, Inc., Australia (2004)
11. Yoo, J.J., Suh, Y.H., Lee, D.I., Jung, S.W., Jang, C.S., Kim, J.B.: Casting mobile agents to workflow systems: On performance and scalability issues. In: Mayr, H.C., Lazansky, J., Quirchmayr, G., Vogel, P. (eds.) DEXA 2001. LNCS, vol. 2113, pp. 254–263. Springer, Heidelberg (2001)
12. Yan, Y., Maamar, Z., Shen, W.: Integration of workflow and agent technology for business process management. In: The Sixth International Conference on CSCW in Design, pp. 420–426 (2001)
13. Savarimuthu, B.T.R., Purvis, M., Purvis, M., Cranefield, S.: Agent-based integration of web services with workflow management systems. In: AAMAS '05: Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 1345–1346. ACM, New York (2005)
14. Moemeng, C., Gorodetsky, V., Zuo, Z., Yang, Y., Zhang, C.: Agent-based distributed data mining: A survey. In: Cao, L. (ed.) Data Mining and Multi-agent Integration, pp. 47–58. Springer, Boston (2009)
15. Fischer, L.: BPM & Workflow Handbook. Workflow Management Coalition (2007)