Journal of Engineering Design and Analysis
Volume 3, Issue 2 - 2020, Pg. No. 19-25
Peer Reviewed Journal

Research Article

Coordination Model for Cross-Organizational Business Process

Tarini Prasad Panigrahi

Gandhi Institute for Technological Advancement, Bhubaneswar, Odisha, India.

I N F O

E-mail Id:
tarinip@yahoo.com
Orcid Id:
https://orcid.org/0000-0003-1175-9728
How to cite this article:
Panigrahi TP. Coordination Model for Cross-Organizational Business Process. *J Engr Desg Anal* 2020; 3(2): 19-25.

Date of Submission: 2020-11-20
Date of Acceptance: 2020-12-03

A B S T R A C T

The mixing of business processes like procurement, finance, human resources and manufacturing in a typical cross-organizational platform is a challenging task. Business process coordination attempts to support the collaboration between different business processes, distributed over different enterprises. In order to coordinate the business processes of different organizations, we need to efficiently integrate different organization's workflows to provide customized servies. Cross-enterprise workflow which can streamline and coordinate business processes across organizations in dynamic Web environment provides a flexible solution. Dynamic web service composition can be achieved by adopting to Service oriented Cross-organizational Workflow (SCW) and Agent Based Architecture. Our purpose is to dynamically integrate the workflows and create a workflow execution community tailored to various workflow specifications. In our work, business processes are wrapped by service agents and each of the agent capability is used to serve a particular service request. In this model based on the customer's requirement, the process agent contacts the service selection agents to find appropriate service agents and then negotiates with the service agents about task execution.

Keywords: Cross-organizational Workflow, Business Process, Service Selection Agent, Process Agent

Introduction

A business process transforms certain input into an output of superior value.[1] Normally it is the preferred output of the customer but it can also add value to the business itself. Whenever a single business process cannot fulfill the needs of an application we combine several business processes in order to fulfill the needs of the application or the customer. These business processes are mapped into web services in a web based environment and the business process coordination is called web service composition. In order to combine the heterogeneous services from different companies we use Service-Oriented Architecture (SOA) platform. Several services during Service composition interact, interoperate, and coordinates so as to achieve the particular goal[2] as needed by an application or customer. Service composition address the issues[3,4] by identifying the tasks to execute of the involved services and the data flow between the services in concert with these roles, and the restrictions on the message's execution orfer (the control flow). Thus, they promote Service orchestration. However the unpredictability and dynamics of various business processes such as faults or breakdown in utility equipment, delays in delivery of raw materials, Production facility failure and cancellation or modification Customer's orders make the business process coordination difficult.[5] For example in a typical supply chain management, whenever there is a change in a customer requirement, then the associates in

supply chain should be instantaneously communicated and respond to the changes accordingly. Thus when an event occurs, it should be immediately propagated throughout the supply chain so as to do timely coordination and interaction among the business processes as per the changes within and between the enterprises. So in designing a supply chain management, the main focus is timely coordination and interaction between various business processes in order to meet the challenge.[6] Here we use workflow system that automates a business process, in whole or in part, during which documents, information, or tasks are passed from one participant to another for action.[7] In order to describe the workflow interaction in the Business Process Coordination, we use the term Service oriented Inter-Organizational Workflow (SCW) when one business process uses the service of another within its own process and is similar to the concept of virtual enterprises.[8,9] Workflow technology becomes a significance technology for business process reengineering. Indeed, an increasing number of organizations have already automated their internal process management using workflows and enjoyed considerable benefits in doing so. However because of the lack of Flexible mechanisms Workflow Management System (WFMS) has had little success in achieving dynamic coordination in SCW. Software agents are autonomous and goal-oriented software entities, which with other agents can operate asynchronously and co-ordinate when needed.[10] Further when we examine a web service, it is a self- describing software process/ component for an application and does not have enough knowledge about its environment, users, software components, and outside world. On the other hand, software agents are capable of reasoning, and interacting with other entities. So, in our approach we use agent based workflow to address the dynamic composition of tasks in cross organizational workflow. Rest of the paper is organized as follows. Section 2, describes our proposed Coordination Model, whereas section 3 discusses how the process agent coordinates with other agents to select a workflow execution plan. In section 4 we continue with our current implementation followed by a scenario of using agent based workflow. Lastly in Section 5 we conclude the paper.

Coordination Model

We use Agent Based Workflow Architecture.[11] as middleware architecture to implement the SCW framework. Figure 1, describes the logical view our agent based workflow. The entire architecture can be classified into three components. They are: workflow definition tool, agent society and actual service. The task of the Workflow Definition Tool is to define cross-enterprise workflow specifications through the end user by using a standard GUI tool. The agent society is a set of agents that forms an agent community which acts cooperatively to provide the general functionalities of a cross-enterprise workflow execution engine. Real Services wraps the applications (called as services) from the physical organizations that allow the user to access and use them to perform the specified tasks during workflow execution.

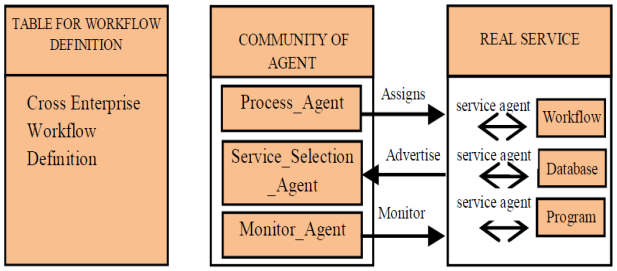

Figure 1.Workflow Management System

This Multi Agent System (MAS) architecture under the control of a workflow management system, used to effectively integrate cross-organization workflow. This MAS consists of a team of software agents, collectively performing a task which could not be performed by any individual agent and further do the negotiation to choose an effective team of agents to complete the tasks of the workflow schema.[12,13] So, having an agent view of the coordination issue in dynamic SIW, we could inherit from numerous concrete solutions projected in this area to deal with agent coordination.[14]

In our Agent Based Workflow Architecture we use five types of agents namely interface agent, service agent, service selection agent, process agent and monitor agent. For each workflow instance these agents form an agent community to execute the workflow. Interface Agent provides the interface through which the user can interact with the Agent Based Architecture system. It submits the XML document for the workflow schema to the process agent to execute the workflow. Service agents are used to abstract the business processes from their physical organizations. Service selection Agent is used to search a specific service in web-based environment which is large and highly dynamic. The service agents advertise and hence register their offerings like identity, capabilities and constraints in the UDDI registry (meta data repository). The meta data is used to locate the service agents. The main task of the process agent is to transform a workflow specification into a workflow instance. It gets the workflow specification from the interface agent and makes a query to the service selection Agent to find the suitable service agents and then integrates the service agents to execute the workflow. A monitor agent monitors the actual execution of a given task at the site of the service provider. In this paper we bring together the dynamic service composition for a set of distributed web services. The SCW environment described in this paper incorporates the interoperability of general web services. Figure 2, illustrates the SCW environment for a typical *Air-condition Manufacturer company Process.*

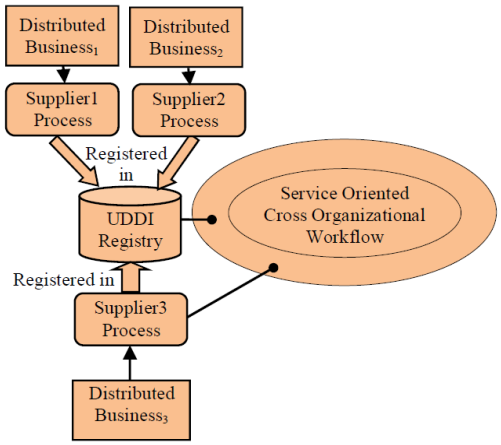

Figure 2.Example of SCW environment

Louis Philippe Ltd. a leading Clothing/ Readymade Garment Manufacturer Company requires different high quality raw materials like Fabric, Dye and Thread used in manufacturing optimum quality Readymade Garments. The company offers a wide range of the finest quality Readymade Garments that includes Ladies Wear, Menswear and Kids Wear those are supplied to various distributors around the globe. Apart from that, the company provides Customized Packaging so as to make the valuable garments free from wrinkles. Thus the company require different raw materials like Fabric, Thread Dye and Packaging from different third party suppliers. (In Figure 2, supplier 1, supplier 2, supplier 3 & supplier 4). The Readymade Garment Manufacturer Company initiates the manufacturing process once it receives the required raw materials like Fabric, Thread and Dye from different Suppliers and then depends on the packaging company to supply the necessary Packaging.

The suppliers for supplying Fabric, Thread, Dye and Packaging register their offerings as web services in a distributed registry, such as a UDDI registry. The manufacturer uses these registry services as a part of its internal workflow. These third party webservices are accessed through the middleware Agent Based Architecture. This agent realization determines the best configuration of agents for the specific cross- organizational workflow process.

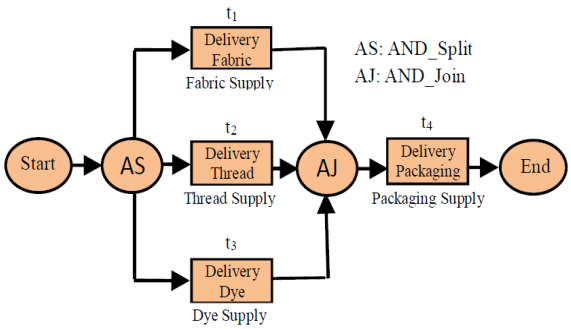

Figure 3.Workflow Patterns For Readymade Garment Manufacturer

Workflow Execution Planning

In order to execute a workflow, the above five agents form a community according to the particular workflow specification. Once the execution of the workflow completes the agent community disbands.

Consider the workflow of the typical Readymade Garment Manufacturer Company, Louis Philippe Ltd, as shown in Figure 3, which requires raw materials like Fabric, Thread, Dye and Packaging from third party suppliers. In order to manufacture the Readymade Garment, the manufacturer company depends on the suppliers: Supplier1, Supplier2, Supplier3 and Supplier4. Once the manufacturer receives the Fabric from Supplier1, Thread from Supplier2 and Dye from Supplier3, then it manufactures the Readymade Garments and finally initiates the Supplier4 process to receive the Packaging-Box.

We start with interface agent submitting the workflow specification to process agent. Further the service agents are used to abstract the business process from the physical organizations and have registered their offerings like identity, capabilities and constraints in UDDI. Then the service selection Agent finds one or more suitable service agents from the UDDI for a given task as per the workflow specification. In this section, we talk about how the process agent coordinates with other agents to pick a workflow execution plan.

Finding the Service Agents

Consider the workflow of the Readymade Garment Manufacturer Company as shown in Figure 3. It involves interaction among agents from a FabricSupply company, ThreadSupply company, Dye Supply and PackagingSupply company. The business workflow has to execute the following tasks:

#DeliverFabric task: Where FabricSupply supplies the Fabric to the manufacturer.

#DeliverThread task: Where ThreadSupply supplies the Thread to the manufacturer.

#DeliverDyeBox task: Where DyeBoxSupply supplies the Dye Box to the manufacturer.

#DeliverPackaging task: Where PackagingSupply supplies the Packaging to the manufacturer.

Using the workflow model defined in our earlier work,[11] the Readymade Garment Manufacturer Company's workflow W can be declaratively defined as follows:

Workflow W = {T, Ew, Rw}, where:

$T= \{t_1, t_2, t_3, t^4\}$

$Ew = \{ew_1, ew_2\}$, where

ew_1: *W → Δt1 Δt_2 Δt_3

ISSN: 2582-5607

e_{w2} :(t_1. result = success) (t_2. result = success)

(t_3. result = success) 4

RW=Null

Then the tasks:

t_1= {N_1, O_1, Et_1, D_1, Rt_1}, where

N_1 is FabricSupply Process

O_1 is DeliveryFabric Process, defined as:

States: {Receive Order, Delivery Fabric, SendInvoice, Receive Payment}

Operations:

Delivery-Item: {Receive Order, Delivery Fabric}

Send-Invoice: {Delivery Fabric, SendInvoice}

Receive Payment: {SendI nvoice, Receive Payment} Et1 = {e11, e12}, where e11:(t1.Fabric. state = ReceiveOrder)

(t_1. Fabric. state = Delivery Fabric)

e_{12}: (t_1.Fabric. state = SendInvoice) (t_1. Fabric. state = ReceivePayment)

D1 = 12 Dec. 2014

R_{t1} = Null

t_2= {N_2, O_2, Et_2, D_2, Rt_2}, where

N_2 is ThreadSupply Process

O_2 is Delivery Thread Process, defined as: States: {Receive Order, Delivery Thread, Send Invoice, Receive Payment}

Operations:

Delivery-Item: {Receive Order, Delivery Thread}

Send-Invoice: {Delivery Thread, Send Invoice}

Receive Payment: {Send Invoice, Receive Payment}

e_{21}:(t_2.Thread.state==ReceiveOrder)⟹
(t_2.Thread.state==DeliveryThread)

e_{22}:(t_2.Thread.state==SendInvoice) ⟹
(t_2.Thread.state==ReceivePayment)

D_2 = 12 Dec. 2014

R_{t2} = Null

t_3 = {N_3, O_3, E_{t3}, D_3, R_{t3}}, where

N_3 is DyeSupply Process

O_3 is Delivery Dye Process, defined as:

States: {Receive Order, Delivery Dye, Send Invoice, Receive Payment}

Operations:

Delivery-Item: {Receive Order, Delivery Dye}

Send-Invoice: {Delivery Dye, Send Invoice}

Receive Payment: {Send Invoice, Receive Payment}

E_{t3} = {e_{31}, e_{32}}, where

e_{31}:(t_3.Dye.state==ReceiveOrder) ⟹
(t_3.Dye.state==DeliveryDye)

e_{32}:(t_3.Dye.state==SendInvoice) ⟹
(t_3.Dye.state==ReceivePayment)

D_3 = 13 Dec. 2014. R_{t3} = Null.

t_4 = {N_4, O_4, E_{t4}, D_4, R_{t4}}, where

N_4 is PackagingSupply Process

O_4 is DeliveryPackaging Process, defined as:

States: {Receive Order, Delivery Thread, Send Invoice,

Receive Payment}

Operations:

Delivery-Item: {Receive Order, DeliveryPackaging}

Send-Invoice: {Delivery_Packaging, Send Invoice}

Receive Payment: {Send Invoice, Receive Payment}

E_{t4} = {e_{41}, e_{42}}, where

e_{41}:(t_4.Packaging.state==ReceiveOrder)⟹
(t_4.Packaging.state==DeliveryPack aging)

e_{22}:(t_4.Packaging.state==SendInvoice) ⟹
(t_4.Packaging.state==ReceivePayment)

R_{t4} = Null

The workflow integrator is used to specify the workflow specification W for the Readymade Garment Manufacturing Company. The Process agent parses workflow specification W, that consists of tasks: t_1, t_2, t_3 and t_4. The process agent queries the service selection Agent for the appropriate service agents which can carry out these tasks t_1, t_2, t_3, t_4. The search result is a set of service agents SPAi. Once the process agent finishes the queries for each task ti in W the process agent gets a set of service agents for each task. Let's assume:

t_1= {spa_1, spa_2, spa_3, spa_{11}, spa_{12}, spa_{13}}

t_2= {spa_4, spa_5, spa_{11}, spa_{12}, spa_{13}}

t_3= {spa_6, spa_7, spa_8, spa_9, spa_{14}, spa_{15}}

t_4= {spa_9, spa_{10}, spa_{14}, spa_{15}}

Or, SPA= {spa_1, spa_2 spa_{15}} (1)

After grouping the service agents by the tasks they can execute, we can have

S*= [<SPA'$_1$, T$_1$'>, <SPA'$_2$, T$_2$'>,[<SPA'$_3$, T$_3$'>,

<SPA'$_4$, T$_4$'>, <SPA'$_5$, T$_5$'> , <SPA'$_6$, T$_6$'>] (2)

Here:

$SPA'_1 = \{spa_1, spa_2, spa_3\}$ and $T_1' = \{t_1\}$

$SPA'_2 = \{spa_4, spa_5\}$ and $T_2' = \{t_2\}$

$SPA'_3 = \{spa_6, spa_7, spa_8\}$ and $T_3' = \{t_3\}$

$SPA'_4 = \{spa_9, spa_{10}\}$ and $T_4' = \{t_4\}$

$SPA'_5 = \{spa_{11}, spa_{12}, spa_{13}\}$ and $T_5' = \{t_1, t_2\}$

$SPA'_6 = \{spa_{14}, spa_{15}\}$ and $T_6' = \{t_3, t_4\}$

Or, in general

S*= [<SPA'1, T1'>, <SPA'2, T2 n, Tn'>] (3)

Negotiation Between The Process Agent and Service Agents

For each task Ti' the process agent starts a negotiation session so as to negotiate with every service agent in SPA'i about executing the tasks Ti'.Our adopted negotiation protocol constitutes 3-steps:

- Request-for-Bid
- Respond-With-Bid
- Counter-Bid

The process of Bidding and counter-Bidding continues till the deadline specified by the process agent. The process agent finally confirms the bid by assigning the tasks to the service agents.

Execution Plan generation

Once the negotiation session is over the process agent can have the bids for different service agents, assigned for different tasks during workflow execution. Now the process agent integrates these service agents to execute the workflow. Here we introduce the following concepts:

Execution Schema

For our workflow W= $[t_1, t_2, t_3, t_4]$, the execution schema(s) can be:

$s = [T_1', T_2', T_3', T_4']$

Where;

$T_1' = [t_1]$, $T_2' = [t_2]$ $T_3' = [t_3]$ and $T_4' = [t_4]$

Here:

$T'_i \cap T'_j = \phi$ where $i \neq j$

$T'_i \subseteq W$

$\sum_{i=1}^{n} T_i' = W$

In this case all the task-sets in s are mutually disjoint subsets and the combination of all the tasks is the workflow W and each Ti' can be executed by different service agents.

Execution Plan

For our execution schema s, one of the execution plans is:

p= [< T1', b1, spa2>, < T2', b2, spa5>, < T3', b3, spa8>, < T4', b4, spa10>]

For every tuple <Ti', ,bi, spai> in p the service agent spai sends bids bi for task Ti'.

The process agent uses the Set Packing $algorithm_{15}$ to generate a set of execution schema(S) from workflow W and S*. Thus, S=[s1,s2,s3 n].

The process agent for each si in S generates a set of execution plan based on the available bids. Thus: P= $[p_1, p_2, p_3\ p_n]$

Here P is the set of all execution plans which can be used to integrate the service agents.

Execution Plan Selection

To select an execution plan the following criteria will be used:

- Cost and Time
- Agent Community Size
- service Agent's Reliability

Execution of Workflow

Once the execution plan is selected, the process agent integrates the service agents to execute the workflow by building an agent community. Here, we assume workflow agent-community has been formed for the workflow specification .Four monitor agents MA1, MA2, MA3, MA4 have been created and deputed to each task's execution site, all the tasks' ECAM rules have been instantiated and downloaded by the monitor agents.

Starting the Workflow

At the start, the process agent will execute the first ECAM rule that is:

$$e_{w1}: *W \Longrightarrow \Delta t_1(MA1) \wedge \Delta t_2(MA2) \wedge \Delta t_3(MA3)$$

The action Δt_1(MA1), Δt_2(MA2) and Δt_3(MA3) results in sending messages to monitor agents MA1, MA2 and MA3 which informs the service agents:

Fabric Supply-agent, Thread Supply-agent and Dye Supply-agent to start executing their task.

Executing the Task t_1 to t_4

Fabric Supply-agent, Thread Supply-agent and Dye Supply-agent begin to execute their tasks after they received the enable signal (Δt_i) essage from the process agent. We assume that Fabric Supply-agent, ThreadSupply-agent supplies the raw materials first. The Object's Item state will be changed to delivered, which will then trigger the action t_3 (MA3) which will send event message (as shown below) to the monitor agent MA3. Now the Dye Supply-agent supplies the Dye and changes the Item's state to delivered.

Message Type: event

Sender: 203.5.1.2:2234: MA1

Event: t_1 (Fabric Supply-agent). Item. state = Delivered

ISSN: 2582-5607

Receiver: 203.5.1.2:1634:MA3

If Fabric Supply-agent, Thread Supply-agent and Dye Supply-agent completes their delivery, the results from the these tasks executions are successful. The monitor agents MA1, MA2, MA3 located at Fabric Supply-agent, Thread Supply-agent and Dye Supply-agent will relay messages to process agent about the their task's state and trigger the dependent task which happens to be task t_4.

The monitor agent continuously pings the service agents in order to check that they are executing their tasks without fail. If a service agent spa (say) fails to execute the task t then the current execution plan is unexecutable and the process agent locate an alternative execution plan to execute the workflow. The process of locating an alternate execution plan involves the following steps:

Step 1: Eliminate all the execution plans which contains the service agent spa's bids from xecution plans.The result is a set of execution plans p (say).

Step 2: Select execution plans (p*) from p such that p*= [< T1', bi, spai>, < T2', bj, spaj>, < T3', bk, spak>, < T4', bl, spal>] and p* is the superset of the execution plan(p) that is just failed.

Step 3: Execute the workflow with such an execution plan in p*. For the tasks which are already finished, replace them with the task results.

Step 4: The monitor agent for this task will update its ECA rules and migrate to the new task execution site. The monitor agent receives completion message from the service agents when they have completed their tasks forwarded it to the process agent. Then the process agent in turn forwards these messages to the interface agent, in order to inform the user about the progress of the workflow execution. Once the workflow execution gets completed the agent community gets released and a new agent community is formed for a new workflow instance.

Implementation of Prototype and Scenario

The prototype implementation of the Agent Based Architecture as in[11] is deployed by using Enterprise Java Beans (EJB). Agent communication channel is implemented using Java Shared Data Toolkit (JSDT). The Service agents and Interface agent are implemented using Java.

We consider the Readymade Garment Manufacturer Company workflow to explain the use of Agent Based Workflow Architecture. The workflow schema consists of four tasks which are Fabric Supply (t_1), Thread Supply (t_2), Dye Supply (t_3) and Packaging Supply (t_4) The workflow has the following specification:

The Readymade Garment manufacturer order for Fabric Supply, Thread Supply and Dye Supply at the same time.

If all these three tasks are successful then the Readymade Garment manufacturer order for the task Packaging Supply.

We created 15-service agents namely spa1, spa2, spa3, ... spa15 all of which registered with the service selection Agent. In the following portion we will describe how to define the workflow schema, and then create workflow instance and then to execute it by using Agent Based Workflow Architecture.

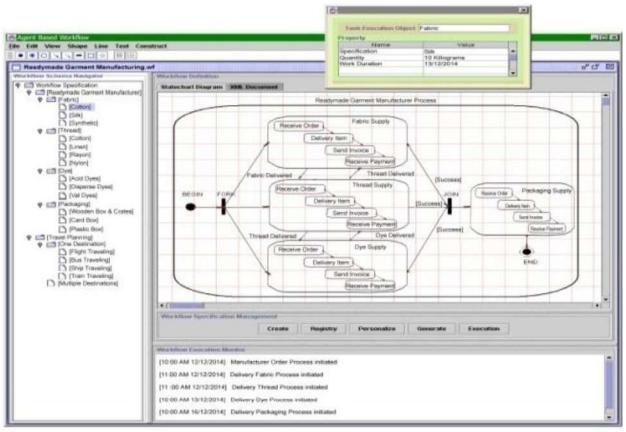

Figure 4.Readymade Garment Manufacturer Company (A scenario)

Description of workflow schema: A workflow panel is provided by the interface agent present in the upper right panel as shown in Figure 4. The process composer with the help of this workflow panel draws the UML state chart diagram that defines the workflow schema. These workflow schemas are organized using domain specific hierarchy. There is leaf and non-leaf nodes in the left panel, the leaf nodes represent the schema of specific workflows. For an instance (as shown in Figure 4) there are two workflow domains namely Readymade Garment Manufacturer and Travel Planning. Fabric, Thread, Dye and Packaging are the sub-domain of Readymade Garment Manufacturer. The Fabric sub-domain in turn has workflow schemas Cotton, Silk and Synthetic. Every non-leaf node in the hierarchy represents a set of workflow schema of a particular domain.

Formation of Workflow Instance

It is possible for a user to access the workflow schema with the help of workflow Schema navigator panel. When the user clicks on the leaf node the respective UML state chart will be displayed on the workflow definition panel. An instance of the workflow can be created by clicking on the create button on the workflow management panel. In order to execute the workflow the user gives the appropriate parameters like the name of task execution object with its specification, Quantity and work duration. Afterwards the user can generate the XML document (that describes the workflow) by clicking on the generate button on the workflow management panel. Once the user clicks the

execution button then the process agent will get the XML document to execute the workflow.

Dynamic Integration

The process agent parses the XML document into three tasks as required by our Readymade Garment Manufacturer process. For each task it queries the service selection Agent for the relevant service agents. Based on the query results from the process agent we separate each group of service agent based on the task-group they are responsible to perform. For example in our Readymade Garment Manufacturer workflow process:

S*= [<SPA'1, T1'>, <SPA'2, T2'>, <SPA'3, T3'>, <SPA'4,T4'>, <SPA'5, T5'>]

Here:

SPA'1 = {spa1, spa2, spa3}

SPA'2 = {spa4, spa5}

SPA'3 = {spa6, spa7, spa8}

SPA'4 = {spa9, spa10}

SPA'5 = {spa11, spa12, spa13}

As there are six set of tasks (T_1', T_2'...., T_6') executed by different service agents in the workflow, we have six different negotiation sections. For our workflow W, the execution schema may be:

S = [s_1, s_2, s_3], where

s_1 = [T_1', T_2', T_3', T_4']

s_2 = [T_1', T_2', T_6']

s_3 = [T_3', T_4' T_5']

Based on the execution schema we can have different execution plans. Let the selected execution plan be: p = [< T3', b3, spa7>, < T4', b4, spa9>,< T5', b5, spa13>] Based on this execution plan the process agent integrates spa7, spa9 and spa13 to execute the workflow of Readymade Garment Manufacturer Company. The agent community will released after the workflow has been finished. A fresh agent community will be formed for new workflow instance

Conclusion

Our aim in this paper is to implement an agent oriented workflow system for dynamic Business process integration. In our approach the agentcommunity for specific workflow is optimally and automatically composed based on the context of workflow execution and can self-adapt and respond to changes during the execution. We show how the workflow agent-community gets constructed by taking the example of Readymade Garment Manufacturer corporation. We demonstrate how the agent community executes the workflow specification. We also used monitor agents for monitoring cross-enterprise workflows. This facilitates the end users to know the status of the workflow instance.

References

1. Hammer M. Beyond Reengineering. Harper Business, 1997.
2. Gustavo A, Casati F, Kuno H et al. Web services. Concepts, architectures and applications. Berlin, Heidelberg: Springer-Verlag. 2004.
3. Barker A, Walton CD, Robertson D. Choreographing web services. *IEEE Transactions on Services Computing* 2009; 2: 152-166. ISSN: 1939-1374.
4. Barros A, Dumas M, Oaks P. A critical overview of the web services choreography description language (WS-CDL). In Proceedings of the business process trends (BPTrends). 2005.
5. Fox M, Barhuceanu M, Teigen R. Agent oriented Supply. *International Journal of Flexible Manufacturing Systems*, 2000; 12: 165-188.
6. Gou J, Yang X, Dai W. On Demand Integration of Dynamic Supply Chain Application Based on Semantic service Oriented Architecture. IFIP International Federation for Information Processing. 254, Research and Practical Issues of Enterprise. *Information Systems* 2007; 589-598.
7. Rl TE. Service-Oriented Architecture Concepts , Technology and Design, Prentice Hall professional. *Technical Reference* 2009; 33-37.
8. Blake MB. B2B Electronic Commerce: Where Do Agents Fit In? ", proceedings of the AAAI -2002 Workshop on Agent Technologies for B2B E-Commerce, Edmonton, Alberta, Canada. 2002.
9. Petrie C, Bussler C. Service Agents and Virtual Eterprises: A survey, IEEE Internet computing, 2003; 1-12.
10. Hess T, Rees L, Rakes T. Using Autonomus Software Agents to Create the Next Generation of Decision Support Systems. Decision Sciences. 2000; 31(1): 1-31.
11. Tarini Prasad Panigrahy and Manas Ranjan Patra; An Agent-Based Model for Cross-Enterprise Supply Chain Management, Transactions on Networks and Communications 2014; 2(4): 107-129.
12. Wooldridge M, Jenning NR. Intelligent Agents: Theory and Practice. In Knowledge Engineering Review, 1995 10(2).
13. Wooldridge M. An Introduction to Multi- Agent Systems. Wiley. 2002.
14. Klusch M, Fries B, Sycara K. Automated semantic web service discovery with OWLS-MX. In: Int. Conference on Autonomous Agents and Multi-Agents Systems, Hakodate. 2006; 915-922.
15. Skiena SS. The Algorithm Design Manual. Springer Verlag, 1997.

ISSN: 2582-5607