ISSN : 0974 - 7435 Volume 10 Issue 20

2014

BioTechnology

An Indian Journal

FULL PAPER

BTAIJ, 10(20), 2014 [12318-12323]

Two implementing ways of workflow systems under the network environment

Li Ya[1,2,3*], Tong Xiong[1,2], Wang Hairui[3]

[1]State Key Laboratory of Complex Nonferrous Metal Resources Clean Utilization, Kunming University of Science and Technology, Kunming, (CHINA)

[2]Faculty of Land and Resource Engineering, Kunming University of Science and Technology, Kunming, (CHINA)

[3]Faculty of Information Engineer and Automation, Kunming University of Science and Technology, Kunming, (CHINA)

E-mail: sherly2001@sina.com

ABSTRACT

Workflow management systems (WfMS) are an emerging category of information systems, currently under the network environment. On the other hand software agents and CORBA objects are two distinct research areas as well as an emerging paradigm for information systems design and development. This paper tries to examine the integration of these two fields. WFMS based on agent technology and CORBA technology can cope with the rapidly evolving business environment better than most other systems as they are more flexible and open. We describe a possible architecture of such a system by means of our prototype WfMS. Collaborating agents or CORBA objects enable a flexible and adaptive system with the possibility of simulation, analysis, and monitoring of the process execution in order to identify potential inconsistencies and to provide appropriate information to the workflow administrator for the purpose of the process improvement.

KEYWORDS

Workflow systems; Network environment; Multi-agent, CORBA.

© Trade Science Inc.

INTRODUCTION

Workflow management systems (WfMS) are systems that define, create and manage the execution of workflows through the use of software, interactions with workflow participants and, where required, invocations of information technology tools and applications (WfMC 1999)[1].

They are typically used in organizations to provide administrative and supervisory functions. Some of the benefits of using a WFMS are:

–Ability to visualize the overall process and interdependencies between various tasks,

– Automation of the processes.

– Automated coordination and collaboration between various business entities.

Existing, commercially available WFMS do not offer sufficient flexibility for distributed enterprises. These systems have rigid, centralized architectures that do not operate across multiple platforms.

–WFMS should find a way to manage the dynamic nature of business processes. As business processes become more volatile, and as they start crossing the organization's boundaries, their interactions need a rather sophisticated supervisor.

–Within business processes, many tasks are interrelated, responsible and data are distributed. This natural concurrency demands efficient techniques for task assignment, resource allocation and scheduling. Moreover, in the case of multiple service providers, the WFMS should be able to semantically discover the appropriate service providers; negotiate with them, and finally allocate them the work.

–Failures and exceptions must be tackled adaptively and efficiently.

–Contemporary WFMS must be able to operate in a pervasive computing environment. They should be able to integrate external applications, other WFMS, heterogeneous devices and legacy systems.

–Operating in the web appears a sine qua non requirement, while supporting the users with friendly and customizable interface would promote their application.

–Scalability, security, and reliability always remain critical requirements.

Without any doubt, there is no single solution for all the WFMS problems and limitations. Moreover, the decomposition of workflows into agent-oriented or based CORBA architectures does not seem an appropriate solution at first sight, since workflows are intrinsically addressed by procedural programs. Let us support this claim with some extra justifications.

One way is using multi-agent technology.

First of all, agents inherit three powerful characteristics from their object-oriented nature: encapsulation, inheritance and polymorphism. In this way agents allow workflow developers to customize workflow objects through subclassing, and improve workflow features through aggregation. Through polymorphism, agents allow to mix and match existing features, dynamically add new features, and adjust the system architecture to a particular domain more easily than any procedural program.

In addition, mobility infuses agents with the ability of migration. This potential allows one to decentralize a WFMS and exploit the benefits of both distributed WFMS and of the agents' paradigm in distributed systems. By their nature, agents support heterogeneity. Using an abstract communication and coordination level, agents can be incorporated into the varying hardware and operating systems architectures that dwell in a business process. This enhanced coordination ability allows agents to act as configuration facilitators and advances them as a promising technology for application integration. Agents' modular nature can provide highly reusable workflow architectures which not only are an alternative technology to existing workflow systems but most importantly, they also offer an alternative vision of how organizations can be structured and managed.

Agents (being autonomous) can relieve workflow engines from some computation. Consequently the engines workloads shall be reduced favoring significantly WFMS scalability. They enable the recovery process as they are stateful entities, contributing significantly to the fault tolerance of the system. The encapsulation of state also supports the asynchronous execution of a business process, a popular case when human participants are involved.

As more general contribution, we may notice that the agent paradigm supports the vision of human substitution: the inherent autonomy of software agents can fulfill activities on behalf of a human with an expected quality of service. Another core feature of agents, reactivity, provides them with an intrinsic capability to adapt to dynamic changes in the environment. Actions do not need to be rigidly prescribed, since agents can timely anticipate their environment as well as efficiently respond to the changes that occur.

Besides reactivity, pro-activeness can boost agents' intelligence. Agents can adopt feedback mechanisms to guide themselves during future actions. They can implement intelligent decision-making techniques such as negotiation, semantics, and planning. Moreover, agents are able to perform optimization tasks as routing and scheduling, task assignment, and resource allocation[2].

Another way adopts CORBA technology.

Many WfMS still have some limitations. First, most such applications are rarely built from scratch; rather they are constructed by composing them out of existing applications and protocols. It should therefore be possible to compose an application out of component applications in a uniform manner, irrespective of the languages in which the component applications have been written and the operating systems of the host platforms. Application composition however must take

into account individual site autonomy and privacy requirements. Second, the resulting applications can be very complex in structure, containing many temporal and data-flow dependencies between their constituent. First of all, CORBA objects inherit three powerful characteristics applications. However, constituent applications must be scheduled to run respecting these dependencies, despite the possibility of intervening processor and network failures. Third, the execution of such an application may take a long time to complete, and may contain long periods of inactivity, often due to the constituent applications requiring user interactions. It should be possible therefore to reconfigure an application dynamically because, for example, machines may fail, services may be moved or withdrawn and user requirements may change. Fourth, facilities are required for examining the application's execution history (e.g., to be able to settle and disputes). So, a durable 'audit trail' recording the interactions between component applications needs to be maintained. Taken together, these are challenging requirements to meet.

In this paper two general designs for WFMS based on multi-agent and on CORBA are proposed. This framework provides a flexible and adaptive infrastructure for incorporating dynamic changes to the currently executing process model. This architecture provides a mechanism for communication of distributed components in order to support interorganizational WfMS.

In Sect. 2 we simply introduce workflow system through CORBA objects. In the rest we put our emphasis on agent technology. In Sect. 3 we give a short overview of the multi-agent based WfMS model which takes the federal style. Section 4 describes the functionalities and the collaborations of our prototype WFMS. We conclude with the results of our research and point out the work which should be done in the future in Sect. 5.

The CORBA Based WfMS Model

We discuss how our system has been designed to meet the requirements stated earlier, namely: interoperability, scalability, flexibility, dependability and adaptability.

– Interoperability: The system has been structured as a set of CORBA services to run on top of a CORBA-compliant ORB thereby supporting interoperability including the incorporation of existing applications.

– Scalability: There is no reliance on any centralized service that could limit the scalability of workflow applications. A dynamic register mechanism of domain engines is introduced to improve the scalability and adaptability of the system.

– Flexibility: The system provides a uniform way of composing a complex task out of transactional and non-transactional tasks. This is possible because the system supports a simple yet powerful task model permitting a task to perform application specific input selection (e.g., obtain a given input from one of several sources) and terminate in one of several outcomes, producing distinct outputs. With an additional route engine, the execution path will be adjusted dynamically according to the execution conditions.

–Dependability: The system has been structured to provide dependability at application level and system level. Support for application level dependability has been provided through flexible task composition mentioned above that enables an application builder to incorporate alternative tasks, compensating tasks, replacement tasks etc., within an application to deal with a variety of exceptional situations. The system provides support for system level dependability by recording inter-task dependencies in transactional shared objects and by using transactions to implement the delivery of task outputs such that destination tasks receive their inputs despite finite number of intervening machine and network failures.

–Adaptability: The task model referred to earlier is expressive enough to represent temporal (dataflow and notification) dependencies between constituent tasks. Our application execution environment is reflective, as it maintains this structure and makes it available through transactional operations for performing changes to it (such as addition and removal of tasks as well as addition and removal of dependencies between tasks). Thus the system directly provides support for dynamic modification of workflows (ad hoc workflows). The use of transactions ensures that changes to schemas and instances are carried out atomically with respect to normal processing[3-5].

The architecture is shown in Figure 1.

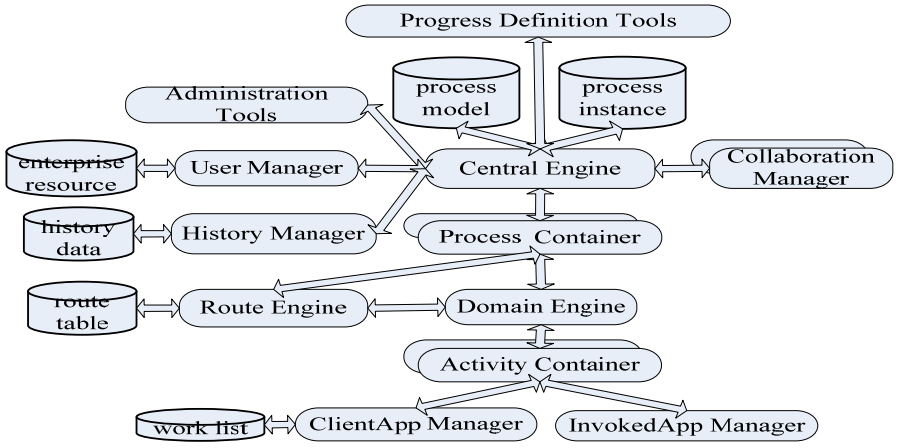

Figure 1 : CORBA based WfMS model

The Multi-agent Based WfMS Model

We design the federal style as specific coordination for MAS. This method has following advantages:

–In the federal style a management agent manages member agents by centralized control mode, to avoid the broadcast communication. This can greatly decrease network traffic, and simplify the structure of agent.

–This style greatly supports the dynamic planning for the aspects of organization structure of workflow management system. It makes integration system to adapt to the dynamic adjustment of business process[6].

The workflow will be considered as the activities completed by several agent federal. The task of workflow management is to control activities to perform coordinately. Our research is focused on developing a multi-agent 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.

The whole system constitutes of an administration agent, a management agent that may take several process agents, a user agent, a resource agent, a history agent, an application agent pool which may take some common agents and some package agents. The high-level model of multi-agent architecture is shown in Figure 2.

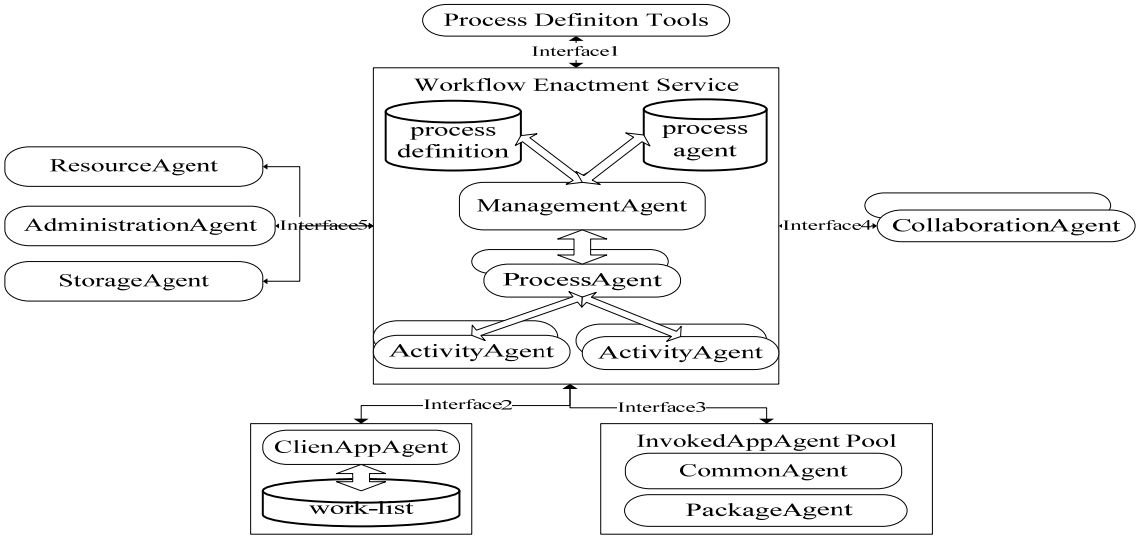

Figure 2 : Multi-agent based WfMS model

Process definition tools have functions of analyze, model, compose, describe, and document a business process. This utility might seem a composite one, but actually the above functions share something in common. These facilities are applied to the process definition during build time. The resulting definition is not operable without agents. The agent language is exploited as a process definition: FIPA protocols are coupled with a process language; agent communication language (ACL) is used to translate the workflow ontology, or agent interactions take place on a speech-acts manner. The internal architecture of agents allows the encapsulation of the process definition.

The whole workflow enactment service is a multi-agent collaborated system. The management agent, the process definition repository and the process agent repository are located on the same host. The process agent which just is an object is established based on according process definition. The management agent is capable and authorized to create, edit, and delete process agent objects within a process definition. They may also edit any of the agent objects' properties. A simple case is to grant permissions to the management agent to access the definition repository. It may create and delete definitions or create and delete process agents. Changes on processes may be applied either on the static definitions, either dynamically, on the executing instance. The management agent may get attributes' values from a specific definition. So it can also retrieve a list of process definitions that fulfill certain criteria and finally, they can retrieve the whole definition itself. When a process definition will be instantiated, the management agent will create a corresponding process agent to be responsible for the control and management of the process implementation, such as initializing a start-up activity agent, collecting status reports of various activity agents and releasing resources when a process instance is completed, etc. As the same, the activity agent is responsible for the control and management of the activity implementation, such as triggering synchronization, start-up task and scheduling follow-up activities, etc.

An agent considers the process as a finite state machine, thus it controls it through state transitions—actions. All actions carried out by an agent are the result of the execution of a certain strategy decided when in a specific state. The decision making abilities of the agent and his strategy selection, eventually provide him with the process control.

Functions of System Agents

Figure 3 Shows these agents of the system and their collaboration

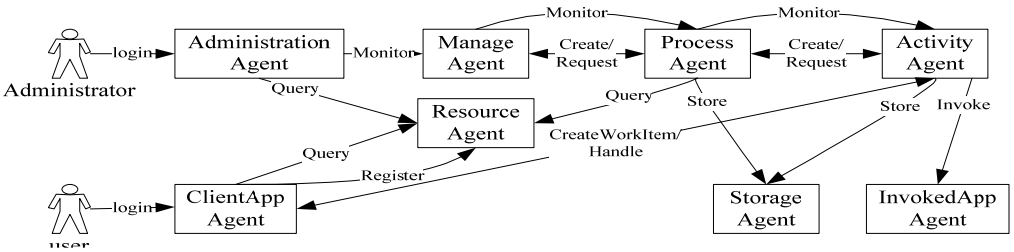

Figure 3 : Function modules of WFMS

Management agent

A management agent supports the runtime environment of a WFMS which refers to a process scope and not to the atomic-task level. Communication among system agents and coordination are the most visible runtime control activities. Additionally it is responsible for managing the process definition repository and the process agent repository, such as import, export and query process definitions and process agents etc. It is able to interpret the process definition language. As more general contribution it also creates, manages and deletes each process agent and supervises the running states of each process instance.

Process agent

A process agent is responsible for the execution of one particular case. For each work case and for each sub work case a new process agent is created. It controls executing process and transiting state of a process instance, such as transmitting sliced model of the process instance according to routing information, and collecting state information of activity instance, etc. It not only provides services of registration, close activity agent, but also obtains relevant information. Every time when adding a new activity agent or start-up an activity agent, the registration service will be called. Accordingly, every time when closing an activity agent, the close service will be called. It controls initiating of its own first activity agent, running activity agents, and state transition. Through cooperation all activity agents in the process can successfully complete the business processes.

Activity agent

An activity agent is responsible for the control and management of the activity implementation, such as triggering synchronization, start-up task, scheduling follow-up activities, and interaction with the client application agent, etc. If the implementation of activity is successful, according to local routing information it will create next activity agent. Otherwise it will inform failure message to the process agent, so that the process agent can trigger a new routing selection for the remainder of process instances.

Client application agent

A client application agent can manage the artificial activity task record, maintain and renew task state. For artificial activities, users must login the system to query and finish his own work.

Invoked application agent

Considering enterprise external applications have various types and interface, in order to reduce the complexity of the calls, all external applications are encapsulated as agents.

Resource agent

A resource agent is the user interface for the human resource or the interface for some tool which can do tasks automatically (such as printers and scanners). The agent represents the resource in the system and negotiates the resource allocation on behalf of the resource and exchanges the necessary information for the execution of a task. It manages system users and its roles, such as user role query, users' logon, exit and recording users' missions, etc.

Storage agent

A storage agent manages the control data of completed processes, and the monitored data. In order to release resources of completed process instances, historical query and statistical analysis conveniently, the control data of completed processes is stored into the historical database by storage agent. It also notifies all management agents if the data has changed.

Administration agent

An administration agent is the management interface of the system. It includes overall and local monitoring and management, such as start-up service, monitoring process instances and activity instances. The administration agent gathers the data in the system that is necessary to analyse workflows, such as execution times and resource utilization. It also gathers

this data associated with a particular case and (after the case is finished) sends this data to the storage agent for persistent storage. It provides the feedback mechanism required for process re-engineering by sensing the anomalies.

Collaboration agent

Collaboration agents are other enactment services component (Workflow interoperability interface). It supplies seamlessly interoperation of workflow systems produced by different vendors.

CONCLUSIONS

In this paper we have constructed two architectures of CORBA and multi-agent based workflow management systems in a distributed environment which provides more flexibility, adaptability and expansibility than existing WFMS. We implemented the described architecture and this paper showed solutions for the problems arising while implementing this system. Our existing framework has been endowed with monitoring and feedback mechanisms, so that the various processes and cases can be studied, analyzed, and the required feedback can be given to the workflow manager. Due to a distributed system, WFMS modular design not only reduces the complexity of the system, but also is easy to expand, as well as has good robustness and reliability. It could support workflow management at the desired level.

As part of our future work will be the integration of sophisticated resource management. One step would be to store all tasks that have been performed by a particular resource to allow more efficient resource scheduling.

ACKNOWLEDGEMENT

This work was supported in part by NSFC under Grant Nos.61263023 and Nos. 51174103. It also subsidized by key projects of Yunnan education department (2012Z107, 2013Z129).

REFERENCES

[1] Workflow Management Coalition, The Workflow Reference Model ttp://www.WfMC.org/standards/docs/tc003v11.pdf, 03 **(2002)**.

[2] P.Delias, A.Doulamis, N.Matsatsinis; What agents can do in workflow management systems, Springer, Artif Intell Rev, **35**, 155–189.

[3] Li Ya, Wang Hairui, Tong Xiong, Zhang Li; Flexible distributed workflow management systems design based on CORBA, Mechatronics and Applied Mechanics, **157-158**, 839-842 **(2012)**.

[4] Li Ya, Tong Xiong, Wang Hairui, Wang Jianying, Zhang Zhibin; Flexible distributed workflow system research, Advanced Materials Research, **569**, 688-692 **(2012)**.

[5] Li Ya, Tong Xiong, Wang Hai-rui; Flexible workflow management through distributed CORBA objects, Information technology journal, **12(19)**, 4992-4998 **(2013)**.

[6] Ya Li, Hairui Wang, Lin Wu; Multi-Agent based design for distributed fault diagnosis systems, Advanced Materials Research, **179-180**, 1266-1271 **(2011)**.