THE RESEARCH OF IMPLEMENTATION METHOD FOR DISTRIBUTED WORKFLOW ENGINES

Luo Haibin and Fan Yushun

CIMS ERC, Dept. of Automation, Tsinghua University, Beijing, P.R.China, 100084

Key Words: Workflow, Workflow Management System, Distributed Workflow Engines

ABSTRACT

A distributed implementation method for the collaborative enactment of workflow engines that run on different enterprise function domains is introduced. The relevant domain concept, the system architecture, and also the interface definition are given. This method not only can overcome the bottleneck of the traditional centralized workflow management system, but also can integrate the workflow with the enterprise modeling process tightly.

1. INTRODUCTION

Workflow technology, a critical technology for modern enterprises to implement the management and control of business processes, has provided an integrated framework for the workflow management from design to running. In a workflow management system (WfMS), a set of integrated and interoperable software tools are offered to support the whole management process. After a decade of evolution, workflow technology has matured to some extent and has been applied to different fields such as telecommunication, banking, health caring, manufacturing and so on. But it is far from its finish. Only a small part of enterprises have adopted this kind of method in their business processes and the area of its application is also limited. The researchers have pointed out the deficiency on system performance, for example, not compatible with each other, not capable of large-scale operations, unstable and insecure [1].

The distribution of workflow engines is a key factor that affects the system efficiency. The coordination of multi-engines which reflects the coordination of people and applications in different organizations will make the system architecture more rational and can easily improve the scalability, reliability and availablity of WfMS. A distributed implementation method for the collaborative enactment of workflow engines that run on different enterprise function domains is introduced in this paper.

2. THE FUNCTION DOMAIN OF ENTERPRISE

When an enterprise is implementing a CIMS engineering, the engineer firstly generates the enterprise function model in the requirement analysis phase from the basic components of the general function model. Along with the evolution of the enterprise business process model, the function model will be refined step by step. In the architecture of CIMOSA, the function view reflects the function structure which includes domain, domain process, business process and enterprise activity. Domain is an important concept in CIMOSA. It represents the new enterprise organizations which are customer oriented and market oriented. The partition of domains will be

based on the goals of enterprise business and the restriction conditions. Different domains can achieve different goals and coordinate each other through events and messages.

According to the concept of enterprise domains and the interaction between the process view and the function view, we integrate the workflow engines, the running core of WfMS, with the process of the enterprise modeling. This can make the build-time system and the run-time system work together tightly. Therefore, we need some new configuration. Firstly, when building an executable process model, associate each basic enterprise activity in the process with proper function domain. This can be done by adding new properties using the process modeling tool. The information of function domains is obtained from the enterprise function model. Secondly, bind each run-time workflow engine with proper function domain. System administrator can do it when starting the workflow engine. Workflow engine in different domain is responsible for different activities, including activating related applications, creating work lists, navigating subsequent activities and so on. Using this method, every task in the enterprise is related with distributed workflow engines. At the same time, the responsibility of every workflow engine is clear.

3. DESIGN OF DISTRIBUTED WORKFLOW ENGINES

3.1 Fragment of Process Model

A problem must be well considered is how the workflow engines get the relevant information of activities when executing them, such as the input and output of the activity, role definition, condition setup and so on. After a completed workflow process definition is built, it will be saved in a central database. In order to make every workflow engine know the activities under control, we divide the whole process into some basic activity units. The fragments will be sent to the proper workflow engine according to their domains and saved as fragment files locally. It is just the fragment file that makes each workflow engine has the executable capability on its own. Also there are some other advantages, for example, (1) database is not required on local node; (2) processing time is short; (3) the activity instance can be easily changed dynamically and the system flexibility is improved.

The fragment file can be described in the format of WPDL or similar standard text. Except for the main body of the activity, the ID number and the domain of the successor should also be recorded so that the local workflow engine can directly activate the next engine to continue the execution of the process instance.

3.2 System Architecture

The system consists of a control workflow engine, some execution workflow engines and a work list manager. The work list manager can be considered as a special execution workflow engine. The control workflow engine is mainly responsible for the maintenance of the configuration information of all the execution workflow engines, creating the process instances, monitoring the status of the instances and so on. The execution workflow engines are responsible for creating the activity instances, invoking the applications, interacting with the work list manager and so on. The work list manager can generate the work items for the manual activity.

The infrastructure that the workflow engines rely on is CORBA. All the workflow engines are appeared as the CORBA objects which are defined by IDL. These objects use IIOP and can receive the invocation from the client embedded in Web. So all the user interface and the administrator

interface are web-based. Figure 1 presents the system architecture and demonstrates a running process instance.

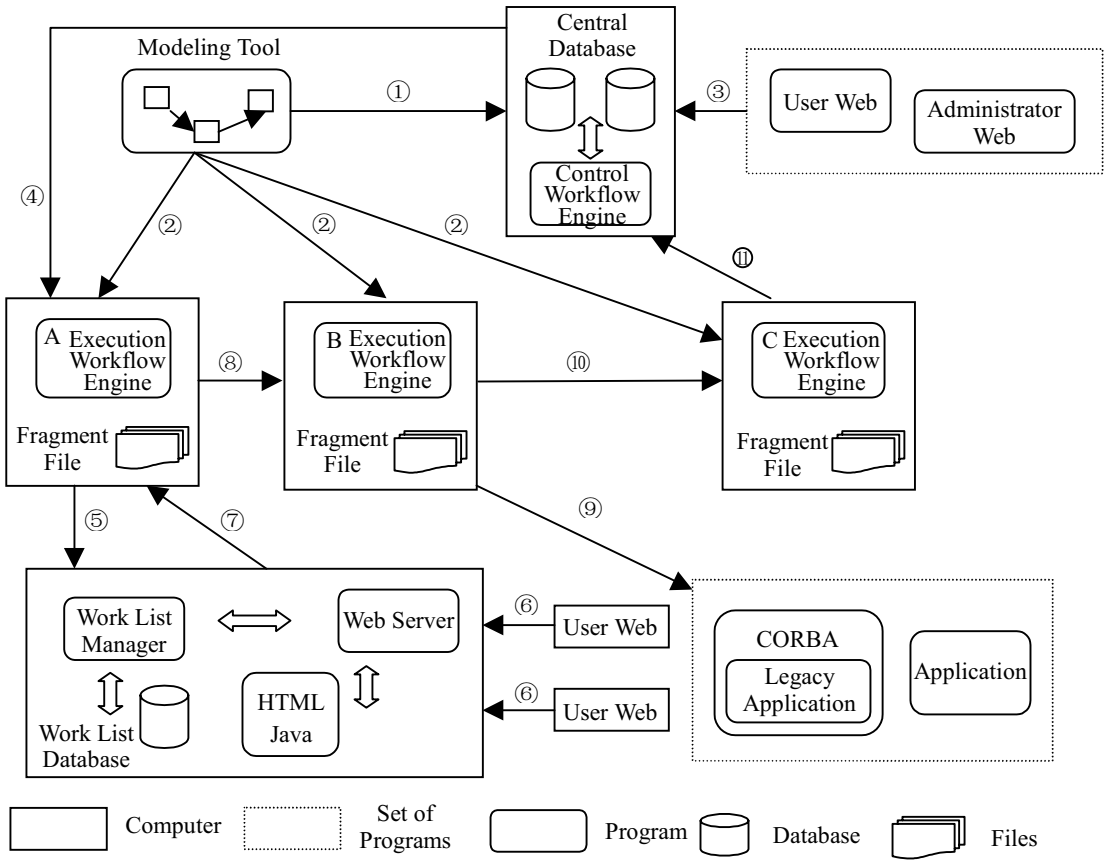

Figure 1. Demonstration of a running process instance

The running steps are detailed as follows: (1) Using the modeling tool, we build a process which consists of 3 activities. The process model is saved in the central database. (2) The process model is divided into 3 parts and each of them is transferred to the execution workflow engine of relevant domain. (3) The process is started through an interface. The control workflow engine creates the process instance firstly. Then the control workflow engine browses the model information in database and gets the IP address of the execution workflow engine which can run the first activity in the process. (4) The interface of the execution workflow engine A is called and the engine A creates the activity instance. (5) The first activity is a manual activity. So the engine A activates the work list manager to create the work items for the activity instance. (6) The workflow participants access their work lists through the web interface and execute the task. After finish it, they submit the task through the same interface. (7) The work list manager notifies the engine A that the activity instance has been completed. (8) According to the model information saved in the local fragment file, the execution workflow engine A notifies the engine B to execute the second activity. (9) The execution workflow engine B invokes other application to finish the second activity which is an automated activity. (10) Similarly, the engine C is notified to execute the third activity. (11) The engine C notifies the control workflow engine that the process instance has ended.

Before the WfMS can provide the enactment service, the workflow engines must start in order. In general the control workflow engine starts at first and then the execution workflow engines start. The enterprise can deploy the engines according to its actual situation.

3.3 The IDL Definition of Workflow Engines

The whole system is designed and implemented using object-oriented method [2]. We use Rational Rose, an object-oriented CASE tool, to define the classes of all the components and generate the IDL files automatically. For the limited length of the paper, we only present the IDL definition of the control workflow engine.

```
interface WorkflowControlEngine {
    //protected member variable:
     ……
    //After binding, "connect" is the first function to call. the clientId is written in log.
     void connect(in string clientId, in string password);
     void disconnect(in string clientId);
    //When some new engine is started or shut down, the two functions are called.
     void registerNewEngine(in CWfBase::EngineInfo info);
     void shutDownEngine(in CWfBase::EngineInfo info);
    //A process instance will run…
     ProcessInstance createProcessInstance(in CWfBase::WMTId processModelId);
    //Other methods for administration usage…
     ProcessInstanceList openProcessInstancesList(in CWfBase :: Filter filter);
};
```

In the interface definition listed above, the CWfBase is a common module which provides the basic data types and exceptions used by other interfaces. In general, the client application obtains the reference of the object in the server and then invokes the corresponding methods to operate the related workflow objects such as the ProcessInstance and the ActivityInstance. The operations on the workflow object include creating and destroying the object, changing the property value, etc.

4. CONCLUSIONS

According to our method, the WfMS will support the enterprise domains in the process definition level and the instance execution level. That is to say, the basic activity unit is bound to the function domain when the workflow is defined and the activities in different domains will be executed and navigated by different workflow engines when the workflow is running. This method not only can overcome the bottleneck of the traditional centralized WfMS, but also can integrate the WfMS with the enterprise modeling process tightly.

Comparing with other distributed WfMSs such as Exotica/FMQM, ORBWork, EVE and DartFlow, our WfMS has some advantages: (1) Fully object-oriented interaction mechanism. (2) Be capable of plentiful and complex business processes. (3) Following the relevant standard of CORBA and WfMC [3]. (4) Friendly and uniform user interface which is web-based.

REFERENCES

1. Alonso G., Agrawal D., Abbadi El A., et al, Functionality and Limitations of Current Workflow Management Systems. Available at URL: http://www.almaden.ibm.com/cs/exotica/wfmsys.ps
2. Booch G., Object-Oriented Analysis and Design with Applications, Second edition. New York: Addison-Wesley, 1994
3. WfMC, Workflow Management Coalition Terminology and Glossary (WfMC-TC-1011). Technical Report, Workflow Management Coalition, Brussels, 1996