APPLICATION ON IOC PATTERN IN INTEGRATION OF WORKFLOW SYSTEM WITH APPLICATION SYSTEMS

Limin Ao [*], Xiaodong Zhu, Wei Zhou
College of Information Engineering, Northeast Dianli University, Jilin, Jilin, China, 132012
[*] *Corresponding author, Address: College of Information Engineering, Northeast Dianli University, Jilin, Jilin, 132012, P. R. China, Tel: +86-432-4807268, Email:aolm@163.com*

Abstract: The integration of the two is a key on introducing workflow technology in the application systems. However, an inappropriate method easily results in invasive system code, structural damage, tight coupling system, and reduces the flexibility of the systems, increases the difficulty to maintain. Then two integration models were presented, and a new integration method was brought forward to solve the problem. Finally, an example was provided to illustrate the implementation, which has been put into practice in the job sheet management system of a power supply enterprise.

Keywords: IoC, workflow, application system, integration

1. INTRODUCTION

Workflow technology is mainly applied to enterprise business process analysis, simulation, definition and the operation implementation, which is the core technology of the realization of business process management and control, process reengineering (WfMC 1995). It can change information processing method of the business and the collaborative approach among the enterprise applications, which has an important influence on the operation and management of enterprises (Zhou et al., 2005). Therefore, the workflow technology has become one of the essential elements on the enterprise information-building program.

But, the current workflow products all have to combine with certain application systems, by which the workflow system can play the role only. However, often due to the diversity and differences of the application, and the improper integration approach, their combination easily results in the use of invasive system code, structural damage and coupling system, and reduces the flexibility of the system, increases the difficulty to maintain.

IoC (Inversion of Control) pattern is a design concept, based on object-oriented basis, to solve the dependence, configuration and life cycle between components, and processing the dependence is its core (Lou et al., 2007). IoC can be used to solve integration-coupling degree of the workflow management system with application systems.

2. INTEGRATION MODEL PRESENTATION

According to the relationship of the workflow management system and application systems, workflow system can be divided into the autonomous and the embedded, two categories (Michael et al., 2005). This paper described the two integration models what has been suggested by Li et al., in 2006.

2.1 Autonomous workflow system integration model

On the interactive mode with the business applications, autonomous workflow engine will provide remote procedure calls (RPC) with the WAPI (Workflow API). In addition, the autonomous workflow engine must also provide the solution how to call the business application.

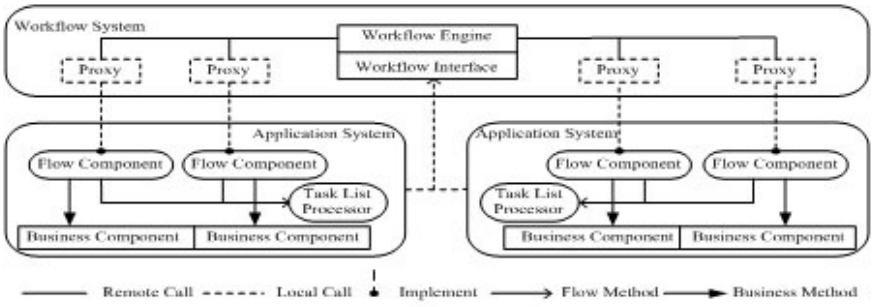

Figure 1. Autonomous workflow system integration model

As shown in figure 1 (Li et al., 2006), the application systems access and drive the workflow engine through WAPI, and the workflow engine triggers and calls applications through the agents that are disposed in the

flow components. But, the workflow interface and proxy mode will restrict the integration capacity.

2.2 Embedded workflow system integration model

Embedded workflow engine doesn't run alone. Generally it is deployed in the application system, as a software component. Therefore, local methods can be called to implement the integration which often is implemented through the two kinds of approach: workflow expansion and component expansion respectively, as illustrated in figure 2 (Li et al., 2006) .

(1) Workflow expansion Business components are the smallest points of function. The flow components that are used to implement the flow agents are directly driven by the engine; Representation layer components directly access the workflow interface. This kind of approach is suitable for flow-oriented system development.

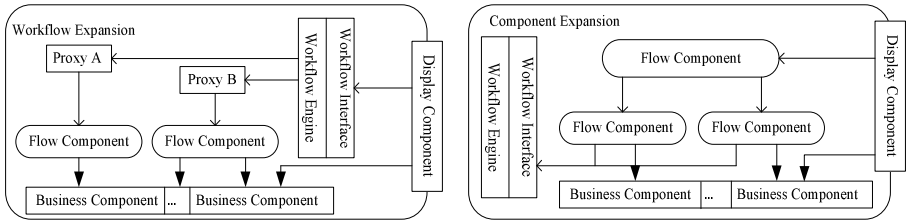

Figure 2. Embedded workflow system integration model

(2) Component expansion Flow component expands the original business components, and implements the flow driver by calling the workflow interface; Representation layer components access the specific flow components. This kind of approach is suitable for increasing workflow management of the existing application system.

3. IOC BRIEF

IoC was brought out by Stefano Mazzocchi, one of the founders of Apache Avalon project. The aim is to emphasize the safety of the design. Hollywood's famous principle: "Don't call us, we will call you" is the embodiment of this idea.

The principle is: components themselves do not actively call the components that are required to complete certain functions, but only declare their interfaces, thereafter inform the superior manager. The superior

manager will provide components that are satisfied to the interfaces. In this way, the dependent relationship will be reversed, and doesn't directly been established any more, instant established in other place. Such, the components won't be hard-coded together any longer each other, then they can obtain the greatest degree of reuse (Tou 2003).

4. INTEGRATION PRINCIPIUM BASED ON IOC

An application is generally composed of a number of components, and those components are interdependent according to a specified rule. In the integration model given hereinbefore, there is the situation of interdependence via calling flow methods or business methods between components. And that relationship of interdependence is represented by hard-coding directly, not through intermediaries. However, such invasive approach will lead to the system tight coupling, lower expandability and maintainability. To a great extent, the components can't be reused, which deviates from software design concept of the "highly cohesive, loosely coupled ". Introducing the IoC will provide the solution of this issue, which can achieve good management of the dependence between components.

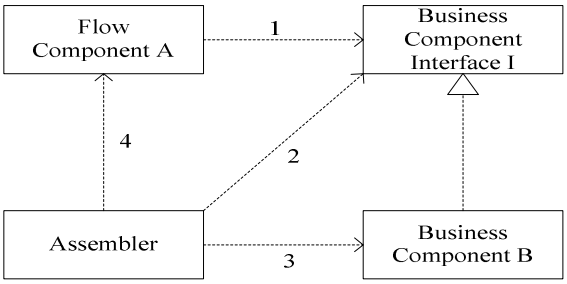

Figure 3. Integration method based on the IoC

According to the principium of IoC pattern, it is known that the essential of solving the component dependence is transferring the dependence. For example, the dependence A--> B between the flow component A and the business component B can be transferred (A->I) + (I<-B) through introducing the inter-protocol and assembler. When the A component calls the method of the B component, at first the I will be declared in the A component, then at runtime the assembler obtains a instance of the interface I according to the configuration file, and injects this instance into the A component, as shown in figure 3. In this way, the component does not actively access the collaboration components or objects no longer, but

dynamically inject ones into the caller by the assembler at runtime (Martin 2004) . This integration method obviously reduces the coupling, avoids a lot of hard coding, is propitious to reuse components and expand systems.

5. SOFTWARE IMPLEMENTATION AND ILLUSTRATION

IoC is a kind of design pattern, whose implementation generally is a container, such as Spring, PicoContainer, and Jdon etc. So, this paper has adopted Java EE lightweight platform, which was composed of Struts, Spring, and Hibernate, and workflow software OSWorkflow to integrate the workflow with applications, at the same time described the example of job sheet management system of a power supply enterprise.

5.1 Integration of lightweight Java EE framework and OSWorkflow

OSWorkflow is a Java workflow engine that has been issued by the well-known open source community OpenSymphony, which adopted FSM (Finite State Machine) theory based on action.

Java EE provides an enterprise-class computing model and the environment, supports the development and deployment of multi-tier architecture application. OSWorkflow is an embedded workflow engine, which can be deployed in the Java EE framework as the part of business logic, the software hierarchy as shown in figure 4.

In this figure, OSWorkflow engine, process components, and business components are all placed in the IoC container, no direct interaction, and their dependence relations are described in the configuration file. IoC container will dynamically inject those collaboration components according to the configuration file at runtime. Thus the workflow system and application systems can be integrated together. Then the degree of coupling can be reduced, synchronously reusability of components also is increased.

5.2 Illustration

In electric power system, the job sheet management system includes the job sheet of No.1 and No.2 of transformer substation etc. Under the above, the dependence relationship of component is incarnated by IoC container. So, the components should be registered into the container at first through

XML desperation file. The example based on the No.1 job sheet of transformer substation is described as follows:

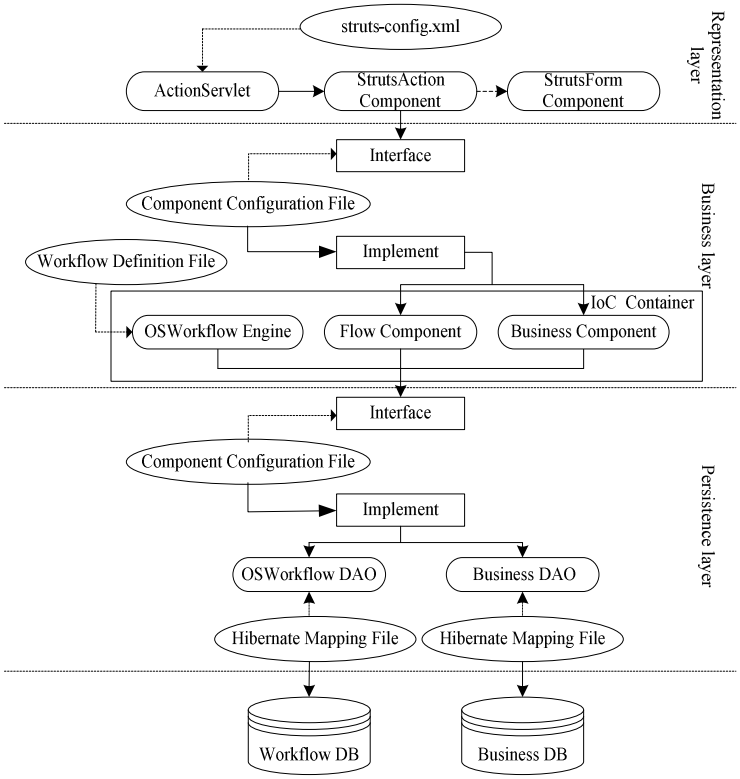

Figure 4. Software hierarchical graph that is embedded workflow engine

```
Registration to Workflow Engine
<bean id="workflow"
        class="org.springframework.aop.framework.ProxyFactoryBean">
      <property name="interceptorNames">
        <list><value>workflowTarget</value>… …</list>
      </property>… …</bean>
Registration to Flow Component
<bean id="flowControl" class="com.mycompony.workflow.FlowControlBean">
  <property name="workflow"><ref local="workflow"/></property></bean>
Registration to Display Layer Component
<bean name="/bdfsBasicInfoAction" class="com.mycompany.struts.action.
        bdfirst.BdfsBasicInfoAction">
      <property name="flowControl"><ref local="flowControl" /></property>
</bean>
```

While registering, the depended components should be declared through the element <ref>. Such, the dependence relationship of component is

transferred to the description file, when changing the dependence, only modify the description file.

Below are the core codes that the representation layer component calls flow components:

```
public class BdfsBasicInfoAction extends Action { ……
public ActionForward execute (...) {.....
    WebApplicationContext wac = WebApplicationContextUtils.
                    getRequiredWebApplicationContext (servletContext);
      // injected by container
        flowControlBean=(IFlowControlBean)wac.getBean("flowControlBean");
      … …}
  … … }
```

In this class, although the representation layer component needs to call the flow components, the latter has not been shown in the former and instant through declaring the interface, and then dynamically injected by the container at runtime, thus avoiding the hard coding.

6. CONCLUDING REMARKS

The key of introducing workflow technology in application systems is to solve the integration coupling degree of the two. Applying IoC pattern in this paper can solve this problem easily. This way cuts down the number of hard coding, greatly improves the reusability of components, makes the whole system flexibility. This solution has been applied to the job sheet management system of a power supply enterprise, which has been showed the availability and practicality.

ACKNOWLEDGEMENTS

This study has been funded by Doctoral Startup Foundation of Northeast Dianli University (BSJXM-200601).

REFERENCES

Li Qing, Wen Jingqian, Zhao Meng 2006. Research on AOP-based integration of workflow system with enterprise information systems [J]. Computer Integrated Manufacturing System,12(3):401-406(in Chinese).

Lou Feng, Sun Yong 2007. Design and Implementation of Lightweight IoC Container [J]. Computer Technology and Development, 17(1):91-93(in Chinese).

Martin Fowler 2004. Inversion of Control Containers and the Dependency Injection Pattern [EB/OL].http://www.martinfowler.com/articles/injection.html.

Michael Zur Muehlen, Rob Allen 2005. Workflow management coalition workflow classification: Embedded & autonomous workflow management systems [EB/OL]. http://www.wfmc.org/standards/docs/MzM_RA_WfMC_WP_Embedded_and_Autonomous_Workflow.pdf.

Tou Ming 2003. Inversion of Control–-Components, Containers and IoC [J]. Programmer, (12):92-94(in Chinese).

WfMC 1995. The workflow reference model[S].

Zhou Jiantao, Shi Meilin, Ye Xinming 2005. State of Arts and Trends Flexible Workflow Technology[J]. Computer Integrated Manufacturing System, 11(11):1501-1510 (in Chinese).