Architectural Design of a Component-Based Application Integration Framework

FUYANG PENG, LELE TANG, XIAOQING WANG
Beijing Institute of Systems Engineering
P. O. Box 9702-19, Beijing 100101
CHINA

Abstract: - Traditional application integration techniques have their problems. Some tools are too complex and vendor-specific, some integration approaches are very ad hoc so that integration developers have a lot of work to do for each component application to be integrated into, and the process is difficult to control. In this paper, we present a component based lightweight application integration framework which is extensible, free from the drawbacks of traditional integration techniques, provides an extensible, reusable and dependable application integration environment for integration developers.

Key-Words: - application integration framework, component-based method, design pattern, enterprise application integration

1 Introduction

The goal of application integration is to integrate the data resources, application systems and business processes within an organization or across organizations to meet new requirements of new application environments, to manage the evolution of component systems, to eliminate redundancy and possible bottleneck, and to resolve mismatches among the component systems [1].

Two main issues are related to application integration [2],[3]. One is how to design a software architectural framework that provides inherent capabilities of and sound development-time/ runtime support for information systems interoperability and integration. The other is how to integrate legacy applications into new software infrastructure or business environment via such techniques as adaptor, mediator and gateway. This is important for the integration and transition of legacy systems.

There exist lots of differences between different integration techniques and tools in terms of integration capabilities. Some are simple and surface, others are very specific, still others are rather comprehensive and powerful.

No matter which is, traditional application integration techniques have their problems [2]. For example, some tools are too complex and vendor-specific, some integration approaches are very ad hoc so that integration developers have a lot of work to do for each component application to be integrated into, and the process is difficult to control. Our goal is, under the guidance of modern software development methodologies, to develop a component based lightweight application integration framework which is extensible, free from the mentioned drawbacks of traditional integration techniques, provides an extensible, reusable and dependable application integration environment for integration developers.

2 Architectural Design of Plumbersoft

2.1 Design Decision

Component-based method and technology is adopted for our message-centric[4] application integration framework called Plumbersoft because of its object-orientation, dynamic loading, component configurability, dynamic typing, serialization and persistence and other features [5,6]. As a result, Plumbersoft is flexible, extensible, reusable and easy to customize. Using Plumbersoft, development time of new integration components(ICs) is greatly reduced, thus integration development efficiency is improved.

2.2 Architecture of Plumbersoft

As we just mentioned, Plumbersoft is a component-based application integration framework. It consists of source binding IC, sink binding IC, transformation IC, message routing controller and system services such as monitor, logging and exception handling. Within the

framework, ICs can be organized into a lattice-like net called p-lattice according to the configuration file provided, standard XML message is used for communication between ICs and message routing controller.

The source binding ICs act as message producers. They implement binding with the to-be-integrated applications, manage the sessions, accept the incoming message and transform it into the common message format, and finally dispatch it into the next-stage IC(s) connected to them via the message routing controller.

Transformation ICs can take some transforming actions to the incoming messages, such as modifying, enriching, filtering and aligning. They have their own abilities. They can be configured. The transformed messages are feed, via the message routing controller, into the next transformation IC or finally into the sink binding IC.

The sink binding ICs are responsible for the binding with the to-be-integrated applications, manage the sessions, transform the incoming message into the external message format or appropriate APIs recognizable by the destination system.

The message routing controller is the key component of Plumbersoft. It selects the routes and controls the transfer of messages. It also coordinates the work of the ICs in the system. Its main functions are:

1. To startup the source binding IC threads and remote controlling thread according to the configuration file.
2. To synchronize with the source binding IC threads and remote controlling thread.
3. To implement global transaction control.
4. To implement the callback registering and triggering mechanism.
5. To implement message routing and control.
6. To manage the ICs and their connections.

More than one p-lattices may exist within an instance of the integration framework. The ICs in the p-lattice may arrange into a linear form or into a more general net-like form. The p-lattices can be executed concurrently.

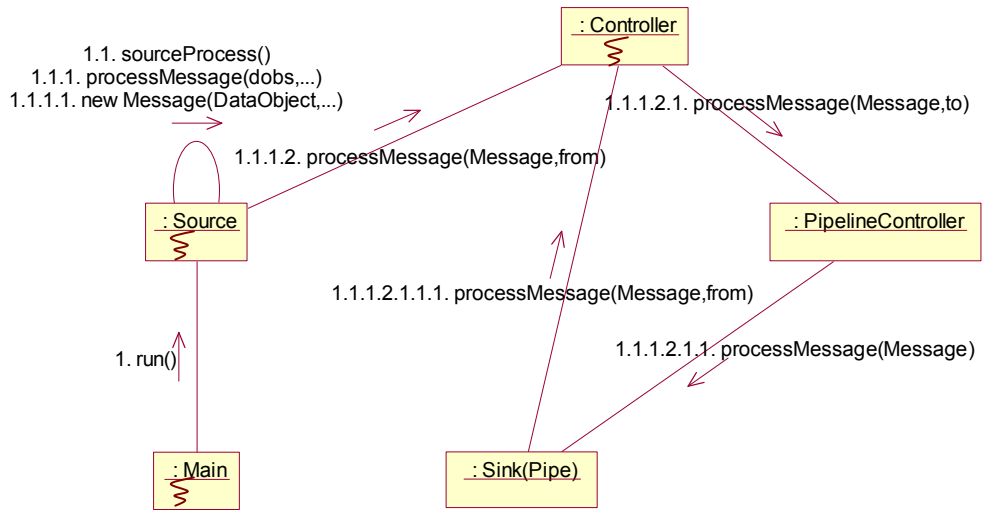

Fig. 1. Activity diagram of the main components in Plumbersoft

Fig.1 is the activity diagram of the main components in Plumbersoft. It is briefly described as follows.

(1) Source binding IC reads into the data objects and encapsulated them into internal common message format.

(2) Source binding IC invokes the message processing method of the message routing controller with the arguments of message itself and the source of the message.

(3) Message routing controller searches for the transformation ICs or the sink binding ICs directly connected to this IC and send the message to these destination ICs via the pipeline controller.

(4) Pipeline controller simply invokes the message processing methods of the destination ICs.

Depending on the type of the destination IC, there are two cases for further message processing.

Case a. if the destination IC is a transformation IC, message is processed as follows.

(5) Invoke transformDataObjects method to take the appropriate transformation to the data objects in the message and form new data objects.

(6) Create new message frame to store the new data objects and push it into message stack.

(7) Delegate the new message to the message routing controller.

(8) Invoke the postMessage method to do some postprocessing.

(9) Pop out the message frame pushed by current IC.

Case b. if the destination IC is a sink IC, message processing is much simpler as following.

(5′) Pop out the message frame from the message stack and read out the data objects. Perform the functions indicated by the message with the help of the destination application's API, such as write the data into a designated file, write the data into a socket or a database system.

2.3 The Use of Architectural Framework and Design Patterns

From the viewpoint of architectural framework, Plumbersoft is an extensible lightweight container[7,8]. We have adopted many design patterns in our design, such as IoC and dependency injection, callback, configurable component, interceptor and singleton[9,10]. All these make Plumbersoft have the following desirable features.

1. The integration framework provides a common methodology and tool for legacy application integration and adaptor development. The framework is flexible, reusable and customizable.

2. The message routing controller, as a container, manages the life cycle of the ICs as well as organizes the ICs into a general p-lattice structure. Furthermore the p-lattices can be executed concurrently. This makes it possible for the complex integration development process to be resolved in a component-based way. Various types of ICs can be developed and categorized. Selected ICs can be configured, customized and combined into flexible work flows for integration purpose.

3. The services in the framework container, such as global transaction support, secure message facility, exception handling, logging, and remote monitoring, make integration more dependable.

3 Conclusions

We have finished the design and implementation of a prototype of Plumbersoft. We have used the prototype to achieve integration of database applications, messaging applications, file-based applications and socket-based applications. Results validate our design. We are taking more experiments, integrating non-Java applications and web services [11]. We are also investigate the possibility of adapting Plumbersoft integration framework to a web service based framework like ESB [12,13].

References:

[1] Rahul Sharma, Beth Stearns and Tony Ng, *J2EE Connector Architecture and Enterprise Application Integration*, Addison Wesley, 2001.

[2] Parker Shi, Suketu Gandhi, "*Enterprise Application Integration*", Centre for Technology Innovation, vol.2 No.3, 2001.

[3] Jeffrey C. Lutz, "EAI architectural patterns", *eAI Journa*l, March 2000, pp.64-73.

[4] Cape Clear Software, "Service-Centric vs. Message-Centric ESBs", *CPV-DOC-3066*, 2005.

[5] S. D. Halloway, *Component Development for the Java Platform*, Addison-Wesley, 2002.

[6] Markus Völter, "PluggableComponent – A Pattern for Interactive System Configuration", *Proc. EuroPLoP '99*, 1999.

[7] Martin Fowler, "Inversion of Control Containers and the Dependency Injection pattern", *http://www.martinfowler.com*

[8] "PicoContainer 1.2 documentation", *http://www.picocontainer.org*.

[9] Erich Gamma, et al. *Design Patterns: Elements of Reusable Object-oriented Software,* Addison Wesley Longman, Inc, 1998.

[10] Douglas Schmidt, et al., *Pattern-Oriented Software Architecture—Patterns for Concurrent and Networked Objects*, Volume 2, John Wiley & Sons, Ltd, 2000.

[11] Steve Vinoski, Integration with Web Service, *IEEE Internet Computing,* November/ December 2003, pp 75-77.

[12] Dave Chappell, *Enterprise Service Bus*, O'Reilly, June 2004.

[13] Mark Endrei, et al., "*Patterns: Service-Oriented Architecture and Web Services*", IBM, April 2004.