World Academy of Science, Engineering and Technology
International Journal of Computer and Systems Engineering
Vol:10, No:10, 2016

Design of a Service-Enabled Dependable Integration Environment

Fuyang Peng, Donghong Li

***Abstract*—**The aim of information systems integration is to make all the data sources, applications and business flows integrated into the new environment so that unwanted redundancies are reduced and bottlenecks and mismatches are eliminated. Two issues have to be dealt with to meet such requirements: the software architecture that supports resource integration, and the adaptor development tool that help integration and migration of legacy applications. In this paper, a service-enabled dependable integration environment (SDIE), is presented, which has two key components, i.e., a dependable service integration platform and a legacy application integration tool. For the dependable platform for service integration, the service integration bus, the service management framework, the dependable engine for service composition, and the service registry and discovery components are described. For the legacy application integration tool, its basic organization, functionalities and dependable measures taken are presented. Due to its service-oriented integration model, the light-weight extensible container, the service component combination-oriented p-lattice structure, and other features, SDIE has advantages in openness, flexibility, performance-price ratio and feature support over commercial products, is better than most of the open source integration software in functionality, performance and dependability support.

***Keywords*—**Application integration, dependability, legacy, SOA.

I. Introduction

THE integration of information systems is to make the data sources, applications and business flows integrate into a coherent entity so as to meet new demands of the complex environment, manage the evolution of component systems, reduce unnecessary redundancies, and eliminate possible bottlenecks of and mismatches among various systems. Any solution of this has to concern with two major issues. The first is to invent a software architectural framework inherently supporting systems interoperability and integration [1]-[3]. The second is for the legacy applications, that is, how to make them integrated into the new framework using techniques like adaptor, mediator and gateway. This is important for the smoth transition of legacy systems [4], [5].

To solve the above problem, a service-enabled SDIE is developed, which gives a common environment for interoperable application development and efficient legacy integration/transition.

The paper is organized as follows. Section II presents the architecture of the SDIE. Section III discusses the dependable platform for service integration, while Section IV describes the legacy application integration tool. Related work is given in Section V. Section VI is the conclusion.

Fuyang Peng and Donghong Li are with the Beijing Institute of System Engineering, Beijing, China (e-mail: fuyang_peng@foxmail.com, dongdong1202@ 126.com).

II. Architecture of the SDIE

SDIE consists of a dependable platform for service integration (DPSI) and a tool for legacy application integration (Plumbersoft). DPSI is the main part of the SDIE, which has four components: a service integration bus, a dependable engine for service combination, a service management framework, and a service registry and discovery module. The service integration bus provides a set of contract communication-based application programming interface (API). It supports multiple workloads, multiple transfer and QoS service, and it also provides a messaging service and security service to make reliable and secure data transfer and message routing possible. The dependable engine for service combination extends the Business Process Execution Language (BPEL) to let dependable requirements defined into the BEPL. It safely combines multiple services into a new service through composition and orchestration. The service management framework encapsulates web services into controlled objects and defines event managing rules. It monitors service states and manages service life cycles to deploy/unload services, initiate/stop services. The service registry and discovery module provides service registry and discovery service. Service registry registers application software and related metadata. Service discovery finds/matches the most suitable service using certain policy and algorithms according to the service registry information.

The legacy application integration tool is a lightweight development and runtime tool for legacy application integration and transition. It is part of SDIE and used for better support of legacy as a service. It can also be used as an independent tool that provides standardized, reusable, customizable, extensible and highly dependable support for application integration development and execution.

III. The DPSI

The DPSI supports three types of user roles, which are the service provider, the service consumer and the service manager. For service provider, it provides support for service description, service registry and publishing, service composition, and dependable design. For service consumer, it provides such functionalities as service query, message transfer and service invocation. For service manager, it provides means for service monitoring and management, QoS guarantee. Loose coupling, flexibility, simplicity and dependability are adhered to when designing the DPSI.

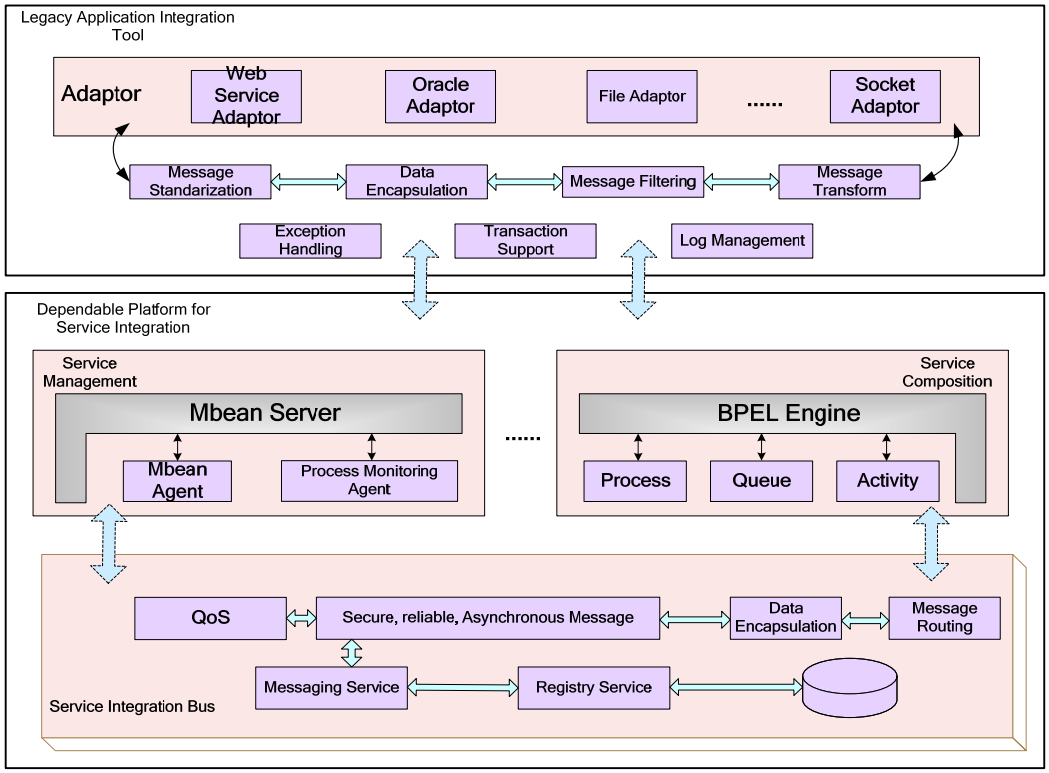

Fig. 1 The Architecture of SDIE

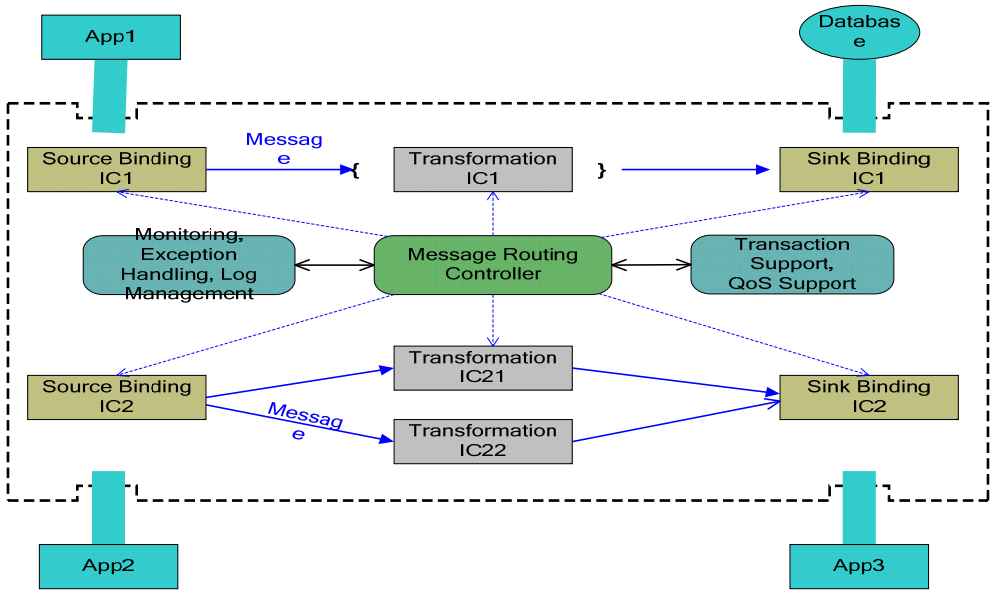

Fig. 2 Basic Organization of the Plumbersoft

A. The Service Integration Bus

The service integration bus provides a set of contract communication-based API. It supports multiple workloads, multiple transferring and QoS service. It provides service-oriented meditation services, such as location transparent service routing and locating, multiple message transfer forms, and various transfer protocols, to support various forms of service integration. It supports multiple interaction models, including uni-directional, request/response, synchronous/asynchronous, and publish/subscribe. The bus is responsible for message routing, event processing and message format transforming. Applications are encapsulated into service forms, and data are transformed into standard SOAP messages before delivering via the bus. It also supports JMS, JCA, JDBC and other commonly used standard interfaces [6]. The bus also provides reliable messaging service and security service to make reliable and secure transfer and message routing possible.

B. The Service Management Framework

The service management framework encapsulates web services into controlled objects with the help of JMX and accesses managed resources using dynamic client proxies. It has an event mechanism to manage the event processing rules. Rules bind events with their processing actions. The framework also provides service life cycle management functions. Two groups of operations are provided to manage and switch the states of the managed services. One is the deploy/unload group that make state switching from initial state to managed state. The other group is start/stop which switches from operational to stop state.

The framework also provides a monitoring service for registered services. It monitors the change of service attributes and triggers notification when change occurs.

C. The Dependable Engine for Service Composition

The dependable engine for service composition is based on BPEL. Connection queue and process state are managed during process execution, which makes flexible configuration and dynamic deployment of service composition process possible. When the engine receives external request, it first finds and locates the required services according to description file, then finds the process instance in the received request queue, and finally decides whether to create a new process or to use the existing process. During execution of the service composition process, the engine takes the service execution sequence of the process as guide and accesses or invokes each service in the process by giving the partners' links, port type, operation name and other optional parameters as inputs.

The engine extends the BPEL to incorporate AOP and Xpath capacity. Dependable requirements can be defined in the extended BEPL, thus dependable service composition can be achieved when dependable service modules are set and triggered during service composition process execution.

D. The Service Registry and Discovery

The service registry and discovery module provides service registry and discovery service. Service registry registers application software and related metadata. Service discovery finds/matches the most suitable service using certain policy and algorithm according to the service registry information. The metadata repository stores the service metadata, configuration information and management rules used in the service integration platform.

IV. The Legacy Application Integration Tool

Plumbersoft, the legacy application integration tool, is designed for the rapid development and convenient runtime monitoring of legacy application adaptors. It supports two types of users. One is the legacy application integration developers and the other is integration users. For developers, it provides support for GUI-style integration configuration, configuration file editing, and integration component customization using the integration framework. For integration users, it provides such functionalities as remote control of integration components and integration messaging diagnosis.

A. The Basic Organization

The core design idea of Plumbersoft is to abstract the common parts of the legacy application adapters into a reusable framework of integration components and abstract interfaces. The framework is based on a component-based, message-centric service architecture, which is flexible, extensible, reusable and easy to customize [7]-[9]. Using Plumbersoft, development time of new integration components (ICs) is greatly reduced, thus integration development efficiency is improved.

Plumbersoft consists of source binding ICs, sink binding ICs, transformation ICs, and message routing controller and system services such as monitor, logging and exception handling. Here we only give a brief description to the Plumbersoft framework. For details, please refer to our other paper [10].

The source binding ICs and the sink binding ICs, acting as message producers and message consumers, respectively, implement binding with the to-be-integrated applications, manage the sessions, and transform the incoming message into the common message format before dispatching it into the next-stage ICs (for source binding ICs) or transform the incoming message into the external message format or appropriate APIs to the destination system (for sink binding ICs). Between the source binding ICs and the sink binding ICs may sit the transformation ICs which modify, enrich, filter and/or align the incoming messages. These ICs can be organized into a lattice-like net called p-lattice according to the configuration, rather than only the simple linear or tree structure. The message routing controller is a service container. It not only manages the life cycle of the ICs but also organizes the ICs into the p-lattice internal form. Its main functions are: to start and synchronize the source binding IC threads and the remote controlling thread, practice global transaction control, provide the event mechanism (i.e. callback registering and triggering mechanism), control the message transforming and routing, and manage the ICs and their connections.

B. Functionalities

Plumbersoft has the following functionalities:

(1) It supports reliable asynchronous data transfer among different application systems or components;
(2) It supports multiple data formats. Different application data formats can be transformed by Plumbersoft to make them speak in a common language;
(3) It provides strong data processing capability. It has built in functionalities of data compression, data encryption and signature, data filling, data format translation and data filtering. It also supports for developers to write his/her own data processing code;
(4) It supports global transaction capability to support consistent commit or rollback operations among different application systems;
(5) It supports data recovery. When failure occurs, it can redo the unfinished operation. Developers also can write their own data recovery code;
(6) It provides large sets of reusable integration adaptors, which may eliminate much of the coding that programmers

have to do before and accelerate the application integration process;

(7) It provides an adaptor development framework to support the development of specific adaptors.

For integration modeling and design, Plumbersoft provides three types of support: editor, GUI, and guidance. Editor mode is the basic, which is suitable for advanced integration developers to directly edit the integration configuration files. GUI mode is easy to use and suitable for both developers and users. Guidance mode is the simplest. One has to follow the instructions step by step to accomplish the integration work.

C. Dependable Measures

Plumbersoft is designed as an extensible lightweight container [11]-[13]. To make our design more dependable, the following measures have taken. Some of the measures are injected into by the container, and others are woven into the p-lattice as transformation ICs.

a) Global transaction mechanism. Plumbersoft provides a global transaction facility to implement the transaction atomicity among multiple applications. The container guarantees that global transaction is committed only after all ICs of the p-lattice are correctly executed, otherwise it will be rollbacked.

b) Exception handling mechanism. Plumbetsoft has taken every possible exception into consideration. It encapsulates the exception handling methods of the to-be-integrated objects into a uniform mechanism so as to make the exception handling more convenient when composing the ICs.

c) Logging and remote analysis. It provides a rich set of categorized logging information, including warning, general, severe, and fatal errors. The logging information can be monitored and analyzed with a remote program.

d) Exceptional Message Mechanism. Plumbersoft provides a message diagnosis mechanism to handle the exceptional messages. The mechanism is actually a temporary message queue that stores the data objects that could not be processed by the adaptors. A special adaptor is designed to read the exceptional messages from the queue and discard the messages or re-route the messages to appropriate ICs according to the configuration parameter.

e) Message Compression and Encryption. Plumbersoft provides transformation ICs for message compression and encryption. Through these ICs, messages can be compressed and/or encrypted during the integration process, so message confidentiality can be ensured.

V. Related Work

Although system software companies, like IBM, Microsoft and Oracle, have EAI products with rich functionalities, these products are usually heavyweight, expensive, and bundled with other products of the company. They have poor flexibility and openness, are difficult to interoperate with products of other companies. SDIE uses lightweight extensible container model, reusable framework and open standard, thus has advantages in openness, flexibility, performance-price ratio.

Many open source software, just taking Mule, Openadaptor, BabelDoc and xBus, as examples, serve the similar purpose to ours. But Mule does not support message container, has no support in configuration, management and monitor frontend, and has poor usability. Openadaptor does not support asynchronous communication. BabelDoc does not support application-based connection, re-composition of message, and message encryption and user authentication. xBus does not support message re-composition and message compression. They have no workflow support, have little support in dependability. SDIE provides good support for synchronous/asynchronous communication, application-based/ middleware-based connection, message container API, message decomposition and re-composition, message encryption and compression and BEPL-based dependable service composition. It also provides user-friendly tools for integration modeling design, integration configuration setting, and monitoring and management.

VI. Conclusions

A SDIE is developed, which gives a common environment for interoperable application development and efficient legacy integration. With the help of the service-oriented integration model, lightweight extensible container, the service component composition-oriented p-lattice structure, and the dependable measures for service integration, both coarse and fine integration granularities are supported on SDIE, and service components on asynchronous WAN can be integrated efficiently, which are impossible in our previous environment. SDIE has advantages in openness, flexibility, performance-price ratio and feature support, comparing with similar commercial products. It is better than most of the open source integration software both in dependable support and in some functional and performance criteria.

Acknowledgment

We would like to thank our team members, Bo Deng, Hailong Li, Lele Tang, Shuangfei Yi and Youyuan Huang, for their sound work and good suggestion.

References

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

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

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

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

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

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

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

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

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

[10] Fuyang Peng and Lele Tang, Architectural Design of a Component-Based Application Integration Framework，*Proc. WSEAS 2007*.

[11] Martin Fowler, *"Inversion of Control Containers and the Dependency Injection pattern"*, http://www.martinfowler.com, accessed on July 5,2015.
[12] Douglas Schmidt, et al., *Pattern-Oriented Software Architecture—Patterns for Concurrent and Networked Objects, Volume 2*, John Wiley & Sons, Ltd, 2000.
[13] *"PicoContainer 1.2 documentation"*, http://www.picocontainer.org. accessed on May 20,2014.