A Best Practice of Enterprise Information Integration Platform in Telecom Industry

Ju-Ting Teng
Billing Information Laboratory
Chunghwa Telecom Laboratories
Taipei, Taiwan, R.O.C.
jt_teng@cht.com.tw

Wen-Chi Wang
Billing Information Laboratory
Chunghwa Telecom Laboratories
Taipei, Taiwan, R.O.C.
emilywang@cht.com.tw

Yung-Chang Tsai
Billing Information Laboratory
Chunghwa Telecom Laboratories
Taipei, Taiwan, R.O.C.
tsaiyc@cht.com.tw

Min-Huang Ku
Billing Information Laboratory
Chunghwa Telecom Laboratories
Taipei, Taiwan, R.O.C.
magicku@cht.com.tw

Abstract—**With the evolution of the digital age, telecom industry proposes a variety of promotional activities to satisfy the market demands, such as fixed line, mobile, broadband access and Internet. However, the systems operate independently without a consistent communication standard. It reduces reusability and development efficiency. In this paper, we propose an information integration platform to expose services, improving the efficiency and stability of integration platform through flow control of service request, integrative monitoring, alarm-notification and single protocol service entry, such as SOAP/WS-SSL, REST/SSL, SOCKET and JMS/MQ. The integration platform provides convergence services with flexibility and reusability. Moreover, our practice releases more than 1000 service interfaces which called for over 300 million times per day. We have improved the bottleneck of interfaces entry. The average response time of each call is less than 1.5 seconds; moreover, the latency time of the integration platform is less than 0.2 seconds. Through the practice, the telecom industry is able to integrate services rapidly and increase company competition.**

Keywords—information integration platform; flow control; monitroing; alarm-notification; service convergence;

I. Introduction

Nowadays, the integration architecture is divided into four types [4]: (1) data level (2) user interface level (3) application interface level (4) method level. Data level architecture relies on transferring data between data stores. In addition, systems extract data from a database; then process and update data in the other databases. In this architecture, the databases might be a bottleneck, and influence the efficiency of transaction. User interface architecture is constructed as a client–server model. Specially, systems have their own interfaces mapping to different systems. In application interface level, interface gives access to services which are provided by the other systems. The main drawback is that systems lack a unique data format and unified interfaces. Method level shares public methods store on a specific location, such as the central server and network.

This paper proposes a practice of enterprise information integration platform, which integrates legacy systems, provides unified interfaces, and supplies a standardized data format for all of the enterprise systems in telecom industry. System resources are occupied by a time-out service interface, while users continually send request. This situation may cause the other services operate abnormally. We present a flow control mechanism to prevent resource lacking by monitoring the legacy systems responses. As the response shows error messages for more than a specific time, the flow control mechanism will take control of the services. Moreover, we provide an integrative monitoring to alarm abnormal service status for maintainers and a single protocol service entry to support multiple protocols requirement.

The strategic objectives of most enterprises are reducing costs, improving quality, and responding quickly to business problems and opportunities. In this paper, we demonstrate an enterprise information integration platform and implement it to prove the forward facility of the architecture. Our research focuses on using industry standards to minimize the architecture's dependence on a particular vendor and enhances flexibility. We also concentrate on flow control mechanism to enhance ability of the integration platform. Furthermore, we will introduce the advanced functions in our future work.

II. Related Works

Enterprise Service Bus (ESB) was introduced as the foundation for enterprise information integration platform, therefore it simplifies the integration of functionalities and services in heterogeneous platforms [2][3]. ESB is a hub for integrating different services through messaging, event handling, and business performance. It supports different kinds of protocols, checks and transforms data format and routes information [1]. Rather than the hub-and-spoke architecture in the past, ESB provides the efficient way to integrate distributed systems, reduces complexity of communication between heterogeneous systems, and adds new services without redesign efforts [3].

This paper puts the ESB into practice in telecom industry and provides a comprehensive service. Moreover, we can support different communications protocols, monitor the integration platform, and alarm maintainers of service interfaces.

III. ENTERPRISE INFORMATION INTEGRATION PLATFORM

A. Platform architecture

We adopt a heterogeneous platform, standard interfaces, modulization and a service-oriented design in the integration platform architecture. The application modules are added into the platform whenever modules are on demand; every module can operate independently. Meanwhile, with unified standard interface, redesign/upgrade modules will not impact on system architecture design.

B. System architecture

The enterprise information integration platform contains systems as: service entry, service integration, monitoring. Service entry system dispatches requests to the proper interface provided by legacy systems. There are several modules in the service entry system which are:

1) Protocol transport: Requester asks for the services through a single protocol, such as SOAP/WS-SSL, REST/SSL, SOCKET and JMS/MQ. To avoid service entry becoming bottleneck, we perform each protocol as a service interface. Requesters are allocated a protocol service instance, and don't need to compete for entering the system.

2) Data transformation: The data of service interface with different formats and encodings are transformed as a Unicode XML format.

3) Access control list (ACL): The ACL files constitute the authority of access interfaces.

4) Flow control: We set an expectant response time $f(M_n)$ with average response time M_n in n'th interface. The value of $f(M_n)$ is represented as a longest response time which the platform is able to tolerate. The service entry has a counter $C(n)$ to record times of the n'th interface response time exceeds expectant response time $f(M_n)$. Equation (1) is the condition that flow control takes place. The threshold counter set by the interface maintainer or manager. During the flow control mechanism taking control of interface, every service request of the interface is rejected by the service entry. The loading of the integration platform is reduced in this way.

$$\text{Flow control while } C(n) > T(n) \tag{1}$$

- n: the n'th interface
- $T(n)$: the threshold counter of n'th interface flow
- $C(n)$: A counter of exceeding expectant response time

$$\text{Time frame}\begin{cases} t_a : \text{L requests access the service} \\ t_b : \text{flow control while C(n) > T(n)} \end{cases} \tag{2}$$

- t_a: time slot in a'th interval, a is odd
- t_b: time slot in b'th interval, b is event
- L: numbers of request, for testing service interface

Recover the operation of interfaces automatically after flow control is a principal issue. We design an interval module to solve the operation recovering. The interval module divides time frame into t_a and t_b (2). In t_a interval, only L numbers of requester are allowed to access the service. The service access is forbidden in t_b interval until a requester gets a success message from service in t_a interval. The flow control mechanism is able to limit requests during exception and auto-detect whether the service is recovered. Once a requester gets a success response, access limit of the service is repealed at once and service turns to normal procedure.

Service integration system provides an integration solution for service-oriented, loosely coupled, highly integrated and widely available protocols.

1) Message routing: Message routing routes and integrates the content for service requesters; therefore, the service requesters are able to ignore the implement of services.

2) Data type transformation: The format provided to the service requesters is unified as XML content. However, legacy systems provide various data formats, such as JSON and XML.

3) Data translation: Data translation translates, cross-reference, and correlates data across multiple incompatible data taxonomies.

4) Database activation: Database activation publishs and subscribes proactively.

5) Work flow: Workflow integrates complex business processes for the service requesters.

Monitoring system covers a real-time service interface monitoring and alarm-notification function applied for service interfaces management in enterprise systems. Functions of the monitoring module are listed below:

1) Message interception and reception: Utilizing UDP to gather message improves the weakness of Request/Reply communication. Collecting message will not affect transference of service message.

2) Message filter: Message filter purifies the message from interception and reception module to reduce memory usage. In the filter, message which has no relation of monitoring are discarded.

3) Message grouping: Messages from the filter module are grouped according to the interface name and the provider (legacy system).

4) Time axis computing: Compute the values of efficiency according to the time condition setting. If the efficiency value exceeds the threshold, monitoring system will send an alarm message to the interface maintainer.

5) Error notifying and alarm: Alarm messages are sent out immediately to notify interface maintainers when exceptions occur. The message is able to be sent as an email or Short message service.

6) Statistics: Statistics function analyzes and collects efficiency information of in 30 minutes. The efficiency information is divided into efficiency of whole system and efficiency of single interface; the statistics chart includes average response time, access success times and access failure times.

7) Cache: To fulfill enquiry of requester and display efficiency information rapidly, the analyzed data is temporarily stored in memory.

8) Log report producing: Log report is created for maintainer reference and inspection. The maintainers are allowed to set procedure of report establishment in monitoring user interface.

9) User interface: The user interface demonstrates the dynamic line chart of interfaces response time and success/error frequency, dynamic pie chart of requester usage factor, Spline chart of service execution time and service exception messages. The service exception messages reveal exception reason, service provider, exception frequency and occurrence date-time.

IV. Statistics Analysis

In this section, a series of real data is demonstrated to evaluate our performance based on the enterprise information integration platform in telecom industry. There are more than 1000 services provided by the ESB and 300 million transactions in the integration platform every day. The call center system requests services through more than 600 interfaces every day. From view of the reusability, the integration platform is the key to save the development cost and the cost of human resources.

There is quite a large amount of communication between systems in the integration platform. The reasons causing long response time are different. A query operation occupies the system resources for a long time, but an application operation spends more CPU and memories. Before we improved our system, there were 12,000 threads in a single server. Due to the limited number of CPU, the switch of threads reduced system performance. We added more servers to get more JVMs and separated the systems with large amount of usage to reduce the conflicts of resources in the peak time. TABLE I. shows the top 5 interfaces of usage frequency in the integration platform on May 14, 2013. The interface of overdue payments inquiry is requested 599,330 times, but the average time is less than 1 second a day.

TABLE I. TOP 5 USAGE OF INTERFACES

Interface Description	**Usage Frequency**	**Average Processing Time(second)**
Overdue Payments Inquiry	599,330	0.147
Get the Device Information	444,489	0.1
Risk Query	335,981	0.036
The Ordering System Profile Query	259,167	0.597
Overdue Payments Inquiry and Risk Detective	205,569	0.164

a. *Usage of interfaces from the enterprise information integration platform* on May 14, 2013

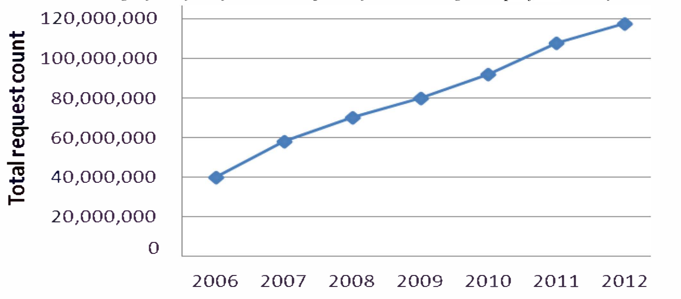

Fig. 1. The growing trend of request count in the enterprise information integration platform

The integration platform integrates all services provided by different systems. Enterprises adopt the integration platform for the efficiency, convenience and flexibility. With the continuous increase in business volume, the platform connects more and more systems. The requirements shown in Fig. 1 rose from 40 million to 120 million in our integration platform between 2006 and 2012.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we propose a practice of an enterprise integration platform in telecom industry, which assures efficiency and quality. We conducted a flow control mechanism to restrict request accesses in service error status. Furthermore, we provide general ESB functions, integration monitoring and alarm-notification services. In addition, every service interface in our system is reusable; the reused amount of existing interfaces is more than 960 interfaces. There are more than 300 million transactions per day, and the latency time of the integration platform is less than 0.2 seconds; moreover, the average response time is less than 1.5 seconds. The statistics exhibit the stability and efficiency of the enterprise information integration platform. Because the integration platform connects numerous systems, our future work is to perform a real-time CRM mechanism by establishing a reaction channel to communicate with customers. The integration platform will no longer lack the decision-making ability.

References

[1] M. Luo, B. Goldshlager and L.J. Zang, "Designing and Implementing Enterprise Service Bus (ESB) and SOA Solutions" in *Proc. IEEE International Conf. SCC'05,* Vol. 2, Washington, DC, USA, pp. 14., July 2005.

[2] J.L. Maréchaux, "Combining Service- Oriented Architecture and Event-Driven Architecture using an Enterprise Service Bus", IBM Developworks, March 2006.

[3] Tibco, (2013) *Enterprise Service Bus* [Online], Available: http://www.tibco.com/products/automation/application-integration/enterprise-service-bus/default.jsp

[4] K. Skalkowski, J. Sendor and R. Slota, "Application of the ESB Architecture for Distributed Monitoring of the SLA Requirements" in *Proc. Ninth International Symposium on Parallel and Distributed Computing ISPDC'2010*, Vol.9, Istanbul, Turkey, pp.203-210, July 2010.