Remote Collaborative Experiments Based on Service-Oriented Architecture(SOA)

Yuen Xing
Mechanical Engineering School
Inner Mongolia University of Science and Technology
Baotou, China
xingyuen@yeah.net

Ercang Yao
China National Nuclear Corporation 208
Baotou, China
yaoercang@yeah.net

***Abstract*—On the basis of analysis for increasingly complex of services logic and functional requirements for the remote collaborative experiments, a new system architecture based on SOA is proposed in this paper. According to the design principles of service-oriented architecture(SOA), different levels and different granularity of services logic and functional requirements for remote collaborative experiments is divided, and a loosely coupled web services system is built. Furthermore, feasible implement methods of key technologies in remote collaborative experiments are given. The benefits of SOA are extended to system architecture: service oriented paradigms are used in the definition of a distributed environment, which allows to hide details about process location, operating systems and programming languages.**

Keywords- Remote Collaborative Experiments; Service-Oriented Architecture; Loosely Coupled; Web Service

I. INTRODUCTION

Some scientific experiments, such as engineering structure and earthquake experiment, are using increasingly sophisticated and expensive facilities. Fewer universities, laboratories, governments and enterprises have all these instruments, equipments or facilities to finish a lager research experiment. Generally, these facilities located in different selected places, and produced by different companies. In this context, experiments, observations and problem solving must be pulled together by scientists and engineers from different locations. Geographic separation becomes a significantly difficulty for accomplishing cooperative experiments. Along with the development of the distributed computer system and high-speed network, computer systems have support the remote cooperative work. The powerful computer can solve the problem of computing capacity and high speed network can facilitate to transmit all kinds of information. And the combination of new information and communications technologies also can compensate the other drawbacks of traditional methods.
The cooperative systems provide a distributed environment for people and instrument cooperation. Information of different kinds (not just data) can exchange, and users in different places can operate the facilities through the remote network(internet or intranet).

In the last days, some remote collaborative experiment [1] systems have been researched, such as NEES in American[2], the Parallel Pseudo-dynamic Testing Systems with the Internet in Japanese[3], On-line Pseudo-dynamic Network Testing in Korea, Internet-Based Simulation of Earthquake Engineering(ISEE) in Taiwan China[4], and NetSLab in China[5]. All these testing systems have specific network transport protocol, interface implement, topology control, program language, and simulation experiment platform. The objects of these remote collaborative experiment systems are to establish research resources network. And then, these resource geographically distributed in various regions can be shared through the network. But it is difficult to share between different system platform because they can not communication and exchange. How to integrate the heterogonous resources which distributed in different places? A large-scale distributed hybrid experiment will utilize a large number of sharing heterogeneous resources (simulation, experimental apparatus), and each owned and controlled by a different institution. How to integrate different simulation platform and physical experiments which can make a collaborative experiment to take place.

In this paper, an new generation remote collaborative experiment research systems is be proposed based on service-oriented architecture(SOA)[6]. And then, we explain how it can ensure secure, reliable, generality in the proposed system. The rest of this paper is organized as follows. Section 2 presents the SOA. The proposed architecture is described in Section 3. Some key technologies are presented in Section 4. Section 5 concludes this paper and future work.

II. SERVICE-ORIENTED ARCHITECTURE

Service Oriented Architecture(SOA) is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. SOA is a flexible set of design principles used during the phases of systems development and integration. A deployed SOA-based architecture will provide a loosely-integrated suite of services that can be used within multiple business domains.

A. Core Idea of SOA

SOA also generally provides a way for consumers of services, such as web-based applications, to be aware of available SOA-based services. For example, several disparate departments within a company may develop and deploy SOA services in different implementation languages; their respective clients will benefit from a well understood, well defined interface to access them. XML is commonly used for interfacing with SOA services, though this is not required.

978-1-4244-6893-5/$26.00

SOA defines how to integrate widely disparate applications for a world that is Web based and uses multiple implementation platforms. Rather than defining an API, SOA defines the interface in terms of protocols and functionality. An endpoint is the entry point for such an SOA implementation.

Service-orientation requires loose coupling of services with operating systems, and other technologies that underlie applications. SOA separates functions into distinct units, or services, which developers make accessible over a network in order to allow users to combine and reuse them in the production of applications. These services and their corresponding consumers communicate with each other by passing data in a well-defined, shared format, or by coordinating an activity between two or more services.

One can envisage SOA as a sort of continuum, as opposed to distributed computing or modular programming.

The following guiding principles define the ground rules for development, maintenance, and usage of the SOA:

- reuse, granularity, modularity, compensability, componentization and interoperability.
- standards-compliance(both common and industry-specific).
- Services identification and categorization, provisioning and delivery, and monitoring and tracking.

B. Service model of SOA

The following service definition focus on the influence of intrinsic behavior in a system and the style of its design:

- Service encapsulation – Many services are consolidated for use under the SOA. Often such services were not planned to be under SOA.
- Service loose coupling – Services maintain a relationship that minimizes dependencies and only requires that they maintain an awareness of each other.
- Service contract – Services adhere to a communications agreement, as defined collectively by one or more service-description documents.
- Service abstraction – Beyond descriptions in the service contract, services hide logic from the outside world.
- Service reusability – Logic is divided into services with the intention of promoting reuse.
- Service composability – Collections of services can be coordinated and assembled to form composite services.
- Service autonomy – Services have control over the logic they encapsulate.
- Service optimization – All else equal, high-quality services are generally preferable to low-quality ones.
- Service discoverability – Services are designed to be outwardly descriptive so that they can be found and assessed via available discovery mechanisms.
- Service relevance – Functionality is presented at a granularity recognized by the user as a meaningful service.

Metadata is used to describe of service in SOA. Metadata description usually contains the following sections: (1) service name, unique identifier of service; (2) message data, the data used to define message types, data structures and exchange patterns between the message; (3) address data, network address of service name is used to address; (4) quality of service, including security, reliability, transaction services strategy, etc.

It is important to divide service description and service implement in SOA. We attempt to establish a loosely coupled web-based system architecture.

III. SYSTEM ARCHITECTURE

In general, entities (people, organizations, and facilities) create capabilities to solve or support a solution for the problems they face in the course of large scale remote collaborative experiments. It is natural to think of one person's needs being met by capabilities offered by someone else; or, in the world of distributed computing, one computer agent's requirements being met by a computer agent belonging to a different owner.

There is not necessarily a one-to-one correlation between needs and capabilities; the granularity of needs and capabilities vary from fundamental to complex, and any given need may require the combining of numerous capabilities while any single capability may address more than one need. The perceived value of SOA is that it provides a powerful framework for matching needs and capabilities and for combining capabilities to address those needs.

A. System requirements

According to some existing studies of the remote collaborative experiments, we can consider that it must meet the following needs:

(1) Compatible the different communication protocols, operating system, program language, hardware platform, and user command format, etc. Furthermore, the friend human-computer interface should be provided by the system.

(2) Shares data among users. The data of the remote environment should be shared by users in some way. If one of them manipulates the shared object, others should be able to receive the latest view. The system should include self-locking. All of them should have the coordination view of the virtual environment.

(3) Provides cooperation mechanism. The user attended in the hybrid experiment should be able to collaborate and interact with others, so they can achieve the remote collaborative experiment efficiently.

B. Remote Collaboritive Experiments

The collaborative environment brings together users, which are geographically distributed, but connected via a network. This not only means that the users will be able to easily communicate, but also collaborate. And the system should be able to keep all the environment data sustained and deliver any change of each user in time. At the same time, the system should be able to deliver collaboration and interaction requirements among the users and apply relevant

mechanism to help them cooperate with each other. The technology behind any synchronous collaboration tool is a mechanism that enables a user to send updates to other users about the interactions that are made in the shared environment.

So, the collaborative system architecture allows geographically distributed users and facilities to cooperate in a large scale hybrid experiment. The architecture is a modular and hierarchy infrastructure to improve reuse, robust, flexibility, interoperability, and scalability. A comparative summary of the different standards for collaborative environments is given in figure 1.

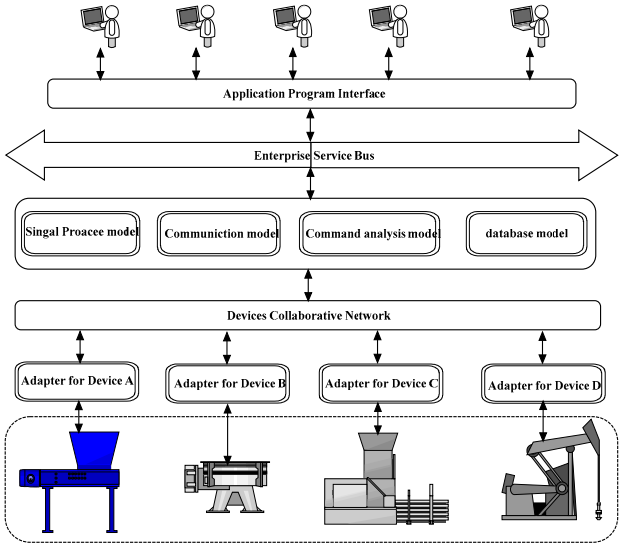

Figure 1. System Architecture

The key part of the whole architecture is the Enterprise Service Bus(ESB) and Devices Collaborative Network (DCN). The ESB is responsible for integration, conversion and analysis. The DCN is responsible for shield the different hardware instruction and devices control.

IV. KEY TECHNOLOGY

A. Enterprise Service Bus

An enterprise service bus (ESB)[7] consists of a software architecture construct which provides fundamental services for complex architectures via an event-driven and standards-based messaging-engine (the bus). Developers typically implement an ESB using technologies found in a category of middleware infrastructure products, usually based on recognized standards.

An ESB generally provides an abstraction layer on top of an implementation of an enterprise messaging system, which allows integration architects to exploit the value of messaging without writing code. Unlike the more classical enterprise application integration (EAI) approach of a monolithic stack in a hub and spoke architecture, an enterprise service bus builds on base functions broken up into their constituent parts, with distributed deployment where needed, working in harmony as necessary.

An ESB does not itself implement a service-oriented architecture (SOA) but provides the features with which one may implement such. An ESB should[citation needed] build on the basis of standards and provide flexibility, supporting many transport media capable of implementing both traditional SOA patterns as well as SOA -enriched business architecture. ESBs attempt to isolate the coupling between the service called and the transport medium. Most ESB providers incorporate SOA principles and allow for independent message formats.

The ESB architecture is as following figure 2.

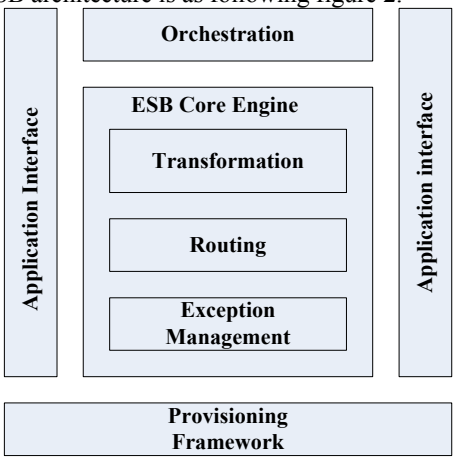

Figure 2. ESB architecture

B. Devices Collaborative Network(DCN)

The DCN will eliminate the difference of hardware devices produced by different company and distributed in different places. As we know, these devices have different control command data format. But the end users maybe not know these data format. Moreover, it is difficult to end users to knowledge all the facilities which included in the remote collaborative experiments. So, DCN can be as an agent to analysis the unified users' operation command and convert commands which input by end user to different devices command format which can be accepted. Conversely, when these devices want to send the feedback, the adapter of devices can convert the feedback data package to prescriptive command format, and the feedback command is encapsulated through the DCN. So, the DCN can agent the device adapters.

C. Other Key Function module

Signal processing is primarily aimed at extracting useful information, while rejecting the extraneous from noisy data. If signal levels are high, then basic techniques can be applied. However, low signal levels require using the underlying physics to correct the problem causing these low levels and extracting the desired information. Model-based signal processing incorporates the physical phenomena, measurements, and noise in the form of mathematical models to solve this problem. Not only does the approach enable

signal processors to work directly in terms of the problem's physics, instrumentation, and uncertainties, but it provides far superior performance over the standard techniques. Model-based signal processing is both a modeler's as well as a signal processor's tool. Model-Based Signal Processing develops the model-based approach in a unified manner and follows it through the text in the algorithms, examples, applications, and case studies. The approach, coupled with the hierarchy of physics-based models that the author develops, including linear as well as nonlinear representations, makes it a unique contribution to the field of signal processing.

Communication model is used to encapsulate all the commands which come form end users or experiments devices. Scheduling of experimentation time is done by an access communication management system. Appointments are stored in a buffer. This is specially important for distance education since some users might choose UNIX-workstations, while others might prefer Windows95/98/NT PCs or MACs. So, the communication model must be compatible different operating system.

Command analysis model is used to encapsulate the command format. Database model is used to unified the different database platform, such as SQL-Server, Oracle, Access, and DB2, etc.

V. CONCLUSION AND FUTER WORK

In this paper, the architecture of remote collaborative experiments based on service oriented architecture is proposed and collaborative mechanisms are described in detail. Furthermore, the key technologies are been discussed. The proposed architecture can provide an effective way to manage the remote collaborative experiment. And the proposed architecture has the characteristic reuse, reliable, commonality, robust. It is advantageous for all users and efficient for all facilities which located in different places. A new solution for remote collaborative experiments has some future work. For example, since we apply Agent technologies in our system, we should consider about the security problem. And another one, perhaps we can also provide users with demo system to validate the feasibility for the proposed architecture.

REFERENCES

[1] J. Pauschke, T.L. Anderson, S. N. Goldstein, et al. Network for Earthquake Engineering Simulation Earthquake Engineering Research Institute. Proceeding of the 7th U.S. National conference on Earthquake Engineering. Boston, 2002:9-19.

[2] K. Lam, R. Butler, I. Foster, et al. NEESgrid System Architecture Version 1.1. Technical report of NEESgrid[R]. USA: National Center for Supercomputing Application, 2002.

[3] P. Peng, T Motohid, N. Masagoshi, et al. On-line hybrid test by internet linkage of distributed test analysis domains. Earthquake Eng Struct, 2005, 34(1): 1407-1425.

[4] D. Park, C. B. Yun, J. W. Lee, et al. On-line pseudo-dynamic network testing on base-isolated bridges using internet and wireless internet[J]. Experimental Mechanics, 2005, 45(4): 331-343

[5] W. K. Jun, L.B. Zhou, C.W. Choung, et al. A platform for networked collaborative structural experiments. Advances in Experimental structural engineering. Japan:Nagoya, 2005

[6] W.T. Tsai, F. Chun, C. Yinong, et al. Architecture Classification for SOA-Based Applications Object and Component-Oriented Real-Time Distributed Computing, 2006. ISORC 2006. Ninth IEEE International Symposium on 2006, 295-302.

[7] M.-T. Schmidt, B. Hutchison, P. Lambros, R. Phippen. The Enterprise Service Bus: Making service-oriented architecture real. IBM report, 2005.

.