Study and Implementation of Spacecraft Integration Test Platform Based on Component Technology

Xianjun Li
State Key Laboratory of Software Development Environment, Beihang University, Beijing, China
Email: lixianjun@nlsde.buaa.edu.cn

Gang Ye, Zhongwen Li, Shilong Ma
State Key Laboratory of Software Development Environment, Beihang University, Beijing, China
Email: {yegang,licw,slma}@nlsde.buaa.edu.cn

***Abstract*—Along with the information construction deeply developed in astronautic field, more and more spacecraft test and experiment systems are constructed. On the one hand, they show their function very well, on the other hand, they bring "information island", which brings more difficulties for the information application integration in aerospace field. On the basis of analyzing this situation, with the component technology, this paper brings out the Spacecraft Integration Test Platform Based on Component (SITPOC), whose aim is to satisfy the new requirements of spacecraft integration test by encapsulating legacy resource and developing new application system in component technology, and describes the function of every layer. Finally, through constructing the integration test database system, the expansibility and the reliability for the integration of the information system in space field are pointed out.**

***Index Terms*—Integration Test Platform (ITP), Component, Connector, Application Integration**

I. INTRODUCTION

In order to improve the work-efficiency and achieve a better management of the test data, each department in aerospace field has put forward their information requirements which adapt to their practical situation, and has constructed these special application systems[1][2].

Although these systems focus on different areas, they have some interactions within each other in function aspect, which causes data inconsistencies, information-sharing difficulties and more garbage information. Meanwhile, it makes it very difficult to implement cross system application. During the process of developing the spacecraft integration test system, this problem is fully aware.

These information systems in aerospace field are basically about testing data management and application, the systems are implemented in a distributed network environment, and the system function basically includes data collecting, data storing, data analyzing and data reporting. And the most important of all is that the data acquisition and processing logical are also great similarities.

Considering these features of information system in aerospace field，this paper introduces a component model technology to encapsulate each logical computing unit as a component, to construct a network system with a set of components and connectors coordinating their functions according to the business model, this technology focus on how to effectively reduce the negative impact of the "information island" in space field, how to build a flexible system architecture, and makes a preparation for the information and digitization in aerospace field.

This paper is organized as follows. Section 2 gives a detailed domain-analysis on spacecraft integration test while section 3 introduces some related technologies. And section 4 brings forward the spacecraft integration test platform which is based on component technology while section 5 defines the composition of the platform. Furthermore, section 6 presents an example of integration test database system to demonstrate the applicability of this platform. Finally, section 7 draws a conclusion for this paper.

II. DOMAIN-ANALYSIS

Spacecraft Integrate test is aimed to test the electrical equipments on spacecrafts and get their parameters of functions and behaviors. In the process of spacecraft integration test, there are some other tests involved. Testers will check the behavior of spacecrafts in various environments by analyzing test data, such as the data of dynamics testing systems and the data of thermodynamics testing systems. However, the distributed professional systems have some data in common, such as parameter definition information and users' information. All these common data should be updated manually when it is needed. Therefore, the inconsistence of crossed systems would be caused easily. The whole structure of the integration test is pictured as follows[2].

doi:10.4304/jcp.6.5.963-968

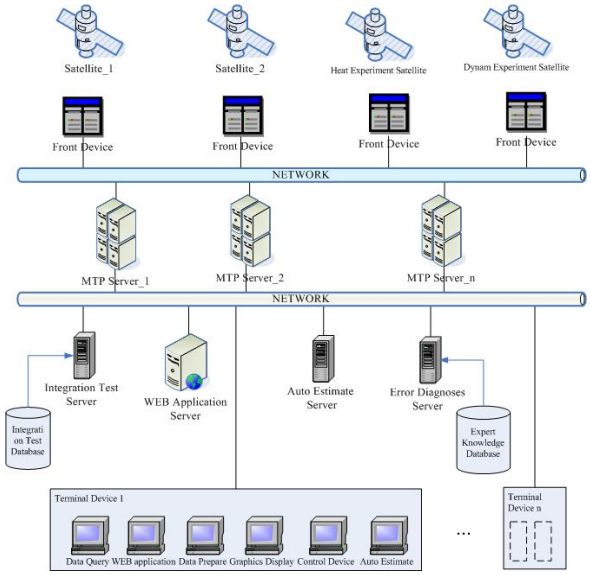

Figure 1. The structure of the integration test

From the figure above, it can be seen that integration test consists of main test processor (MTP), integrated test database system, automatic read-check and failure analysis system and many other existed systems. These systems, which are deployed in distributional places, cooperate with each other to accomplish the complete integration test process, and provide functions including test data storage, data query and analysis, remote monitoring, automatic error check and failure diagnosis.

In order to normalize the process of spacecraft test and accomplish the unified management on the test data, there should be an application system integrated platform. This platform not only supports the current distributed application, but also meets the requirements of extension and stability.

III. RELATED TECHNOLOGIES

Focus on solving the problems of isomerism between different application systems, Corba, J2EE and COM/DCOM provide great middle-ware environment platform for distributed computing. However, these technologies and criterions do not give a unified definition and management on data resources. Therefore, there is no good to the far-reaching requirements of enterprise information process. Moreover, these technologies and criterions are general solutions to various fields so the huge organization architecture and communication mechanism can not be used to specific requirements of spacecraft test system.

A great many of research have focusing on the definition of component and architecture[4][5], each of them are too comprehensive and are not exactly compatible to the requirement in spacecraft test system. Other study and applications focused on special field such as C2 architecture which reorganized the components in a layered model to resolve GUI designing issues[6], definitions of components in astronautic field should also be refined to fit to be configurable.

The development of component-based software development technology and enterprise application integration technology provides solutions to information construction[7][8][9]. The main difference between component technology and traditional software development technology is that component technology reuses the existed components to set up new application systems. This process is showed in Figure 2.

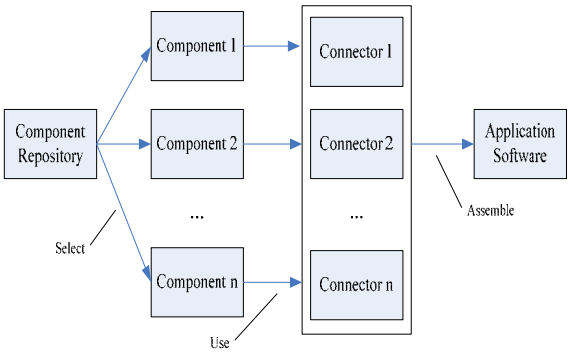

Figure 2. The process of software

Enterprise application integration can be divided into three levels: data integration, enterprise business integration and application system integration. It can integrate data sharing and business process without too much modification on existed application. It makes users access different application system transparently. The idea of enterprise application integration can be seen in the following figure 3.

Based on these technologies, integrated test system can be divided into different functions each of which can be encapsulated into different components. Using this technology, business logic is separated from data as well as application. All components which obey to some kind of criterions can be assembled into a new application system and achieve the goal of reusing the existed application system when developing new applications.

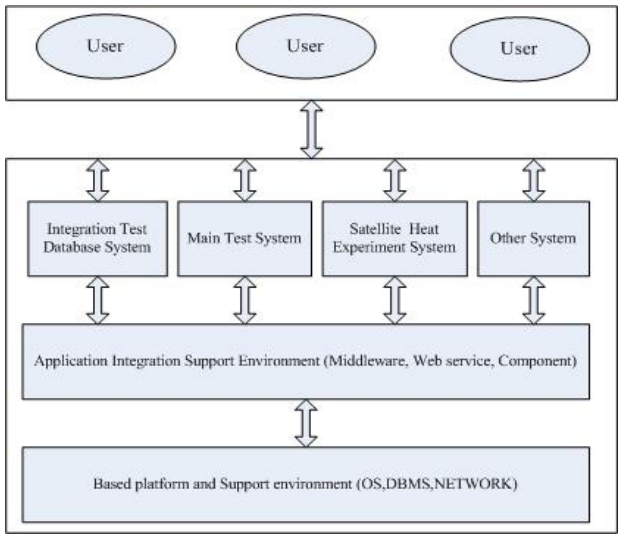

Figure 3. The idea of enterprise application integration

IV. SPACECRAFT INTEGRATION TEST PLATFORM BASED ON COMPONENT

According to the technologies of component and enterprise application integration[10][11][12], on the functional requirements and extensible requirements of the spacecraft integration test, a new application integration platform is advanced to satisfy the requirements of spacecraft integration test.

The platform is showed in figure 4. This platform consists of data infrastructure platform, component support platform and application platform. The functions of every layer are described as follows.

Data infrastructure platform: this layer provides the function of unified data inserting management, data interface management, unified data pick-up, data resources catalog management. This layer is aimed to transform data resources in different types, eliminate data isomerism and unify data management by using a certain defined data interface criterion. And on the basis of that, it provides unified data service for the upper layer.

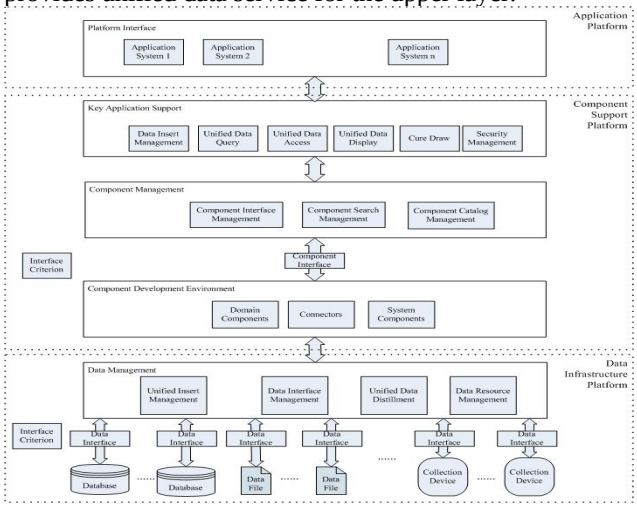

Figure 4. spacecraft integration test platform based on component

Component support platform: this layer is composed of component development environment, component management environment and key operation support. The operation component, connect component and the system component, which are the result of operation logic modeling, are developed in the component development environment. Component interface management, component search management and component catalog management are included in the component management environment. They provide the function of unified management of component and search, utilization of the component. Key operation support, on the principle of function-independence, assembles the component into some application systems which have some certain operation functions.

Application platform: On the support of some visually integration tools, this layer provides the function of integrating the key operation support and related component into an application system.

V. COMPOSITION OF THE PLATFORM

A. Definition of component

Spacecraft components are the most basic element of composing the test application platform, what the platform can complete the complex business application function were all achieved through collaboration of these relatively simple, basic business functions components. So, in system integration level, which emphasis on concern is the component can provide the service as well as its business functions to be called by other component services, and shielding the concrete code realization situation. Its realization may is a process of code, also may be class, classtree, frame, or by several types of composition of modules, and they can even be a computing system[13]. In general, as long as it can be clear identified and be reused by some software system, can be called component. The minimum size for a single component is classes and modules.

Accessing from the outside, a component can be seen as a set of computing services. It is the basic element and minimal unit for designing the system architecture. The set of services can be divided into two categories, respectively are components themselves can provide the business functions to external service interface collection and the need to call a collection of other components of the service interface.

Since the spacecraft applications run in a distributed real-time environment, we need complete description of the component in a distributed real-time environment from the following several aspects[14], service functions (the interface name of the components can provide the business function service for the external system), service call parameters, the service returns results. As members of the services provided, taking into account whether the real-time characteristics of certain services, if any, need to consider access to these properties of real-time service period of time. Summing up the above factors, the component can be defined as follows.

component ::= {name, providedServices, requiredServices};
name ::=componentName;
providedServices ::= service, providedServices;
requriedServices ::= service, requiredServices;
service ::= {servicesName, parameters, result, realtimePriority, realtimePeriod};
servicesNames := string;
parameters := (parameter, parameters) | null;
parameter := paramtype, paramname;
result := returntype;
realtimePriority := 0|1;
realtimePeriod := int;

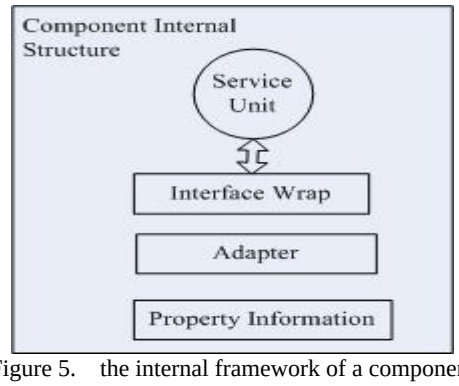

Figure 5. the internal framework of a component

As showed above, a component consists of four parts:

Service Unit: the implementation of computing services which is provided by the component. It is the really different part from one component to another.

Interface Wrap: wrapping outside service unit to define the services and providing different methods to invoke the computing functions of service unit written in different programming languages.

Adapter: the low-level communication medium through which the component exchanges data with the outside. Adapters send and receive messages from the connector connected to the component in different ways according to the type of connection between the components.

Attribute Information: It is the description of the component, which gives some aids to assemble a system or to select a component.

B. Definition of connector

Connector is the bridge joining components together by enabling them call the services from each other. Each connector server is used for one single component to provide services and is connected by other requesting component which would call for these services. Moreover, since the type of service calling between components could be classified as calling in the LAN environment and invocation within a single thread, the connector has two corresponding types, remote connector or local connector. Remote connector is the connection between component service call in the local area network environment. Each remote connector corresponds to a service component, receives the service requests, manages all requests and returns the results to the service caller. The remote connector played a role in routing, which will link the distributed components in the network transparently together, collaborate to complete a business function. Because the multiple instances of a same service component might be deployed in a remote host, remote connector searches position of a component instance only through the IP address and port number of the remote component. It uses Socket technology to achieve messages communication between remote connector and service request component and service response component

Local connector is the connection between components in the local same process. This kind of mutual service call of components is in the single process in a same host, thereby can eliminate the need for component route in network distributed environment.

Therefore, the formal definition of the connector is as follows:

connector ::= remoteConnector|localConnector

remoteConnector ::= {name, rmSvcComponent, rmReqComponents, constraints }

name ::= connectName

rmSvcComponent::= {componentName, address, port}

rmReqComponets::= {componentName, address, port}, rmReqComponets

address ::= ipAddress

port ::=portNumber

constrains ::= {ipConstrains, ipServiceConstrains}

ipConstrains ::= allowedIPSet

allowedIPSet ::= ipAddress，allowedIPSet

ipServiceConstrains ::= {ipAddress, allowedService},ipServiceConstrains

AllowedService ::= serviceName, allowedService

localConnector ::= {connectName, lcSvcComponent, lcReqComponents}

lcSvcComponent ::= componentName

lcReqComponents ::= componentName, rmReqComponets

The connector also has interface, its interface is composed by a group of roles. Each role of the connector defines the connector participants. Dual connectors have two roles, such as message transfer connector is the role of the news sender and the news receiver. Some connectors have more than two roles, such as event broadcast in one event publishers and any more events recipients' character role.

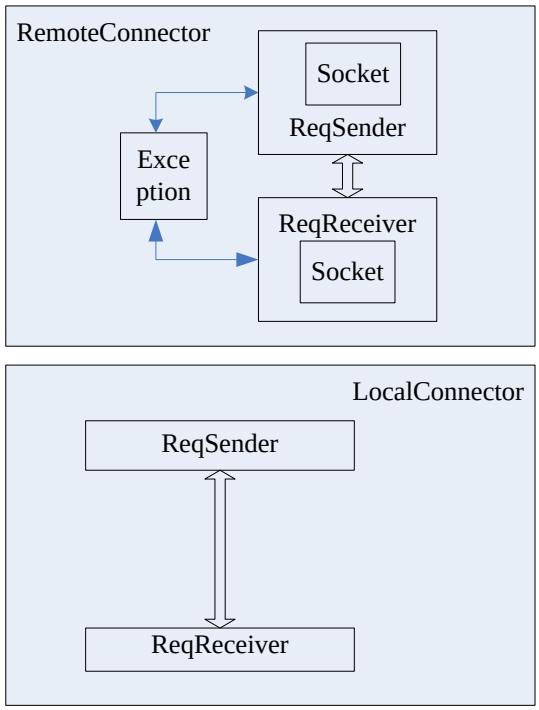

Figure 6. the internal architecture of the connector

The internal architecture of the connector is showed in fig6. The RemoteConnector is composed of three component, ReqSender class, ReqReceiver class and Exception class. Whereas LocalConnector is composed only by two component, ReqSender class and ReqReceiver class, which does not exist exception information processing of network communication.

C. Connect mechanism

Application system is composed of components, connectors and the corresponding constraints. Connectors need to complete the function of receiving service call request message, sending service request message, receiving the results of service calls and returning the result of service call, and so on.

RemoteConnector is responsible for connecting message services between the components in a distributed network environment. It use the Socket technology to communicate with the components RemoteAdaptor and achieves the message communication between the service request component and the service response component.

LocalConnectors implement the message service of the components through a function call parameter passing and returns the results. The service call components connected by LocalConnectors are all in the same process and in the same host, hence the implementation of the service call message and result transferring between the local components through the function call and results return of the components, and the function call and results return of the connectors.

D. Application system assemble

Application system could be defined as a collection of three elements including the collection of components, the collection of connectors and the links among all components and connectors.

The formal definition is given below.

Architecture = {components, connectors, linkages};

components = {component, components};

connectors = {connector, connectors};

linkages = {linkage, linkages};

linkage = {reqComponent, connector, svcComponent, services};

services = svcComponent.service, services;

VI. TRAIL APPLICATION

We now take the implementation of Integration Test Database System(ITDS) as a concrete example to demonstrate how to assemble application system on this platform. The ITDS, which is an important part of the platform, accomplishes the function of data receiving, data storing, data querying and real-time data monitoring.

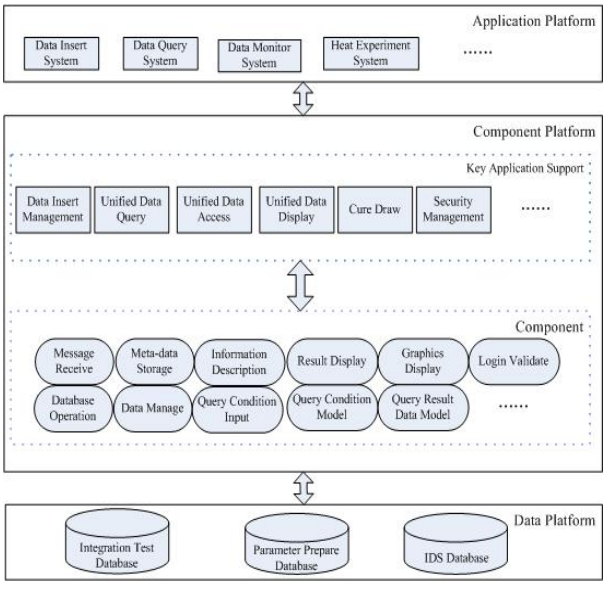

Figure 7. The structure of integration test database system

Therefore, in terms of the framework of the platform, according to the principle of function-relatively-absolute, we divide ITDS into three sub-system, including data-insert sub-system, data-query sub-system and real-time monitoring sub-system. They are deployed in a distributed network environment. On the basis of the platform, components are connected by connectors; they are assembled to be a application system which have some special functions.

The ITDS has been successfully used in a certain type of spacecraft test, practical applications show that the system has better performance, universal and expansibility.

VII. CONCLUSION

This paper analyzes in detail the state of the information construction in aerospace domain, points out the main problems, and then according to the function requirements of the spacecraft integration test, brings forward the spacecraft integration test platform based on component, which is aimed to satisfy the application requirements of the spacecraft integration test. Furthermore, the key elements of the platform are defined and the assembling mechanism is advanced. Finally, the applicability of the platform is demonstrated by constructing the integration test database system.

During the process of implementation, it is proved enough that using component technology can efficiently reduce the system development cycle, efficiently integrate current legacy system resources and date resources, efficiently figure out the problems that the information island brings.

In the future, the run-mechanism of the platform should be studied more, and more tools should be provided to support assemble and integrate application system.

REFERENCE

[1] Requirement definition of satellite’s dynamics experiment system

[2] Requirement definition of satellite’s heat experiment system

[3] Requirement definition of satellite’s integrate test system

[4] Li Qingsheng,Wang Jipeng,Wang Aimin. Application of Component Technology in Multimedia Integration System. 2009 First International Workshop on Education Technology and Computer Science P1100-1103

[5] Shifeng Zhang, Steve Goddard, An Architecture Description Language to Specify Component-based system, Proceedings of the international Conference on Information Technology: Coding and computing (ITCC’05)

[6] R. Taylor. A component- and message-based architectural style for GUI software. IEEE Transactions on Software Engineering, 22(6):390--406, June 1996.

[7] Xia Cai, Michael R.Lyu, Kam-Fai Wong, Component-Based Software Engineering: Technologies, Development Frameworks, and Quality Assurance Schemes, 2000 IEEE.

[8] D.S Rosenblum and R. Natarajan, Supporting architectural concerns in component interoperability

standards, IEE Proceedings online no.20000913, IEE, 2000
[9] DASHOFY.E.M,MEDVIDOVIC.N, and TAYLOR.R.N, Using off-the shelf middleware to implement connectors in distributed software architectures, 21st international conference on software engineering, May 1999,Los Angeles, pp3-12
[10] M. Radestock and S. Eisenbach, Component Coordination in Middleware Systems, IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing Lancaster. U.K., 1998,P225-240
[11] Kotonya, G.; Sommerville, I.; Hall, S.; Towards a classification model for component-based software engineering research. Euromicro Conference, 2003. Proceedings. 29t , 1-6 Sept. 2003 Pages:43 - 52
[12] D.Batory and S.O'Malley. The Design and Implementation of Hierarchical Software System with Reusable Component. ACM Transactions on Software Engineering and Methodology.1(4):355-398,October 1992
[13] Peter Hans Frohlich.Component-Oriented Programming Languages:Why,What,and How[D].University of California.2003
[14] Sun Guotan. Research and Implementation of Component Based Development for Spacecraft Experiment Systems[D]. Beijing, Beihang university.2005

Xianjun Li (1977 -) is currently PhD candidate in State Key Laboratory of Software Development Environment, Beihang University, Beijing, China. His research interests include spacecraft automatic testing, spacecraft integrate test information Processing, component technology.

Gang Ye (1982 -) is currently PhD candidate in State Key Laboratory of Software Development Environment, Beihang University, Beijing, China. His research interests include spacecraft automatic testing, spacecraft integrate test, software testing，fault localization.

Zhongwen Li (1981 -) is currently PhD candidate in State Key Laboratory of Software Development Environment, Beihang University, Beijing, China. His research interests include workflow, spacecraft automatic testing, realtime system.

Shilong Ma (1953 -) is Ph.D. professor, doctoral supervisor, standing deputy director of the National Key Lab for Software Development Environment of the School of Computer Science and Engineering in BUAA, member of the 10th expert appraisal panel under Department of Information Science of the National Natural Science Foundation Committee, member of Executive Committee of Asian Software Fundamental Federation, standing director of China Artificial Intelligence Society. His research interests include Grid, Computational Logic, Magnanimity Information Processing, spacecraft automatic testing.