Distributed System Based on Publish-Subscribe Middleware Automated Testing Method

Xingmin Wang¹, Meng Yang² and Peng Han³

¹ Product Development Center, China ShipBuilding IT CO., LTD, Beijing, 100861, China
² Product Development Center, China ShipBuilding IT CO., LTD, Beijing, 100861, China
³ Product Development Center, China ShipBuilding IT CO., LTD, Beijing, 100861, China

*Corresponding author’s e-mail: 414106503@qq.com

Abstract. Publish-subscribe middleware has been widely used in military, medical and other important industries because of its good characteristics of time and space decoupling. With the continuous expansion of the scale of distributed system application, the problems in distributed system testing, such as low efficiency and difficulty in fault positioning, are more prominent. In this paper, an automatic testing method is proposed, which is based on the characteristics of the publish-subscribe middleware systems. Based on an independent designed and implemented publish-subscribe system named BlueDCS, a corresponding automatic testing tool is developed. Application verification shows that the tool can well solve the aforementioned problems in distributed system testing.

1. Introduction
Publish-subscribe middleware based on publish-subscribe message paradigm realizes the decoupling of space and time between publisher and receiver, and provides multi-level service quality guarantee for information interaction in distributed system. Publish-subscribe middleware designed and implemented based on Data Distribution Service (DDS), a publish-subscribe middleware specification formulated by OMG, has been widely used in military, medical and other industries [1][2].

With the expansion of the scale of distributed systems and the complexity of their functions, how to quickly and automatically complete system testing and locate faults with the support of publish-subscribe middleware has become an urgent problem to be solved. At present, there is no effective automated testing tool for publish-subscribe middleware. The joint debugging and integration testing of distributed system still need to be completed by testers manually, which seriously affects the development efficiency of distributed system.

One of the main tasks of distributed system testing is to test whether the components in distributed system run normally or not. Four kinds of conditions for judging whether the components in publish-subscribe distributed system run normally or not have been summarized, meanwhile, a component automated testing method suitable for publish-subscribe middleware systems has been proposed on this basis in the paper. By means of publishing and publish-subscribe middleware, the running state information [3] of components in the system can be obtained by the method, and whether the pre-set system testing conditions are satisfied or not can be judged through the analysis of the state
information, to realize the automatic test of the system. An automated testing tool for publish-subscribe distributed system has been implemented on the premise of self-developed and DDS-compliant publish-subscribe middleware BlueDCS based on the method in the paper, and the effectiveness of this method and tool has been verified in practical application.

In the second part of this paper, four kinds of conditions have been summarized to judge whether the components in publish-subscribe distributed system run normally or not, the automatic testing method of publish-subscribe distributed system based on monitoring mechanism has been expounded; In the third part of this paper, the design and implementation of the automation testing tool for publish-subscribe distributed system have been introduced; In the fourth part of this paper, the full text has been summarized and the next step has been put forward.

2. Testing Conditions and Automated Testing Method in Publish-Subscribe Distributed System

2.1. Testing Conditions in Publish-Subscribe Distributed System
The purpose of distributed system testing is to test whether the components in the system can work properly in the environment provided by the system, that is, to observe whether the components are running as expected in the actual system. In publish-subscribe distributed system, the core actions of components are the publication and subscription of data, and the frequency and sequence of actions are the basic measures to judge whether components work normally or not; In addition to the action of publication and subscription of the components, the correctness of data content is also an important indicator to determine whether components work properly or not. In addition, components running in distributed systems also need to meet system constraints on component resources.

Four basic conditions of component testing have been summarized by means of the synthesis of aspects involved in component testing in the paper, they are action consistency condition, data legitimacy condition, module response time condition, and resource constraint condition respectively.

2.1.1. Action consistency conditions
Action consistency conditions describe the sequence of publication and subscription of data of components and represent the logical relationship of data among components. Action consistency conditions can be simple or compound consistency conditions. Simple consistency condition expresses the sequential relationship between two component actions, and compound consistency condition is composed of several simple consistency conditions through logic and/or relationship. Simple consistency condition structure is < Pre-component, Pre-component action, Pre-relationship identifier, Post-component, Post-component action >, and the definitions of each part are as follows:

- Pre-component
  Pre-component represents the component that acts first in the simple consistency condition. Its structure is (component declaration keyword ["Comp"], component name).
- Pre-component action
  Pre-component action represents the specified action of the pre-component in the simple consistency condition, which needs to occur before the corresponding action of the post-component. Its structure is (action type, action corresponding topic), action type is publication, expressed as "pub", or subscription, expressed as "sub". The action corresponding topic is the theme corresponding to publication and subscription of data.
- Pre-relationship identifier
  Pre-relationship identifier is a binary operator that indicates that actions on the left need to occur before actions on the right. Its symbol is "->".
- Post-component
  Post-component represents the component that generates the response action in the simple consistency condition. Its structure is the same as the pre-component.
- Post-component action
Post-component action represents the response actions generated by post-component, which should occur after the specified action generated by the pre-component. Its action is the same as the pre-component.

Examples of action consistency conditions are as follows:
Concurrent: Comp A pub Topic α & Comp B pub β & Comp A sub Topic δ & Comp C sub Topic θ

The enumerated action consistency condition are logically connected by two simple consistency condition, in the simple consistency condition (Comp A pub Topic α & Comp B pub β), pre-component is component A, pre-component action is the data with publication theme α, and post-component is component B, post-component action is the information with publication theme β. The second simple consistency condition indicates that component A subscribes to data with topic δ before component C subscribes to data with topic θ.

2.1.2. Data legitimacy condition
The correctness of data content published or subscribed by components is also an important indicator to measure whether components work properly. Data legitimacy condition provides a definition of the expected values for the various contents of data. Its structure is < Data legitimacy condition declaration keyword, Object component, Topic, Attribute, Value range >, each part is defined as follows:

The declaration keyword of the data legitimacy condition is "Content", and the object component refers to the component that needs to meet the data legitimacy condition. Topic refers to the topic dominated by the data legitimacy requirement, the attribute is the attribute targeted by the data legitimacy requirement, and value range refers to the expected value range.

Example of data legitimacy condition: Content: Comp A Topic θ attribute β limit: 10 - 20, the object component is component A, what is required for data legitimacy requirement is the attribute β in the topic θ, and the value range of β is 10 to 20.

2.1.3. Module response time condition
Module response time condition, a module, or a component or several components or even the whole system, refers to the requirements of the time from receiving information from the outside of the module to making the specified response. The scope of influence of this condition varies with the size of the module it aims at, which can be used for both global testing and fault determination of individual component. Module response time condition structure is <Module composition, Signal, Response action, Time limit>, each part is defined as follows:

- Module constitution:
  Module constitution specifies which components the module consists of, and this part explains the influence range of the response time condition of the module. The structure is (module keyword "module", module name, component list). Module keyword is "module"; Module name is a string composed of letters; The list of components is framed by "["", in which the component names involved are written separated by spaces.

- Signal
  Signal refers to the specified action that triggers the response of the module. Generally, it is a subscription or publication of data on a specified topic. Its structure is as follows: (Signal definition keywords, Signal source, Action type, Topic). Signal definition keyword is "signal". Signal source refers to the component that generates a signal. Action type refers to subscription or publication of data. Topic is the topic involved in action.

- Response action
  Response action refers to the response action produced by the module after receiving the signal. Its structure is < Response action keyword, Responder, Action type, Topic >. Response action keyword is "action". Responder is the component that generates the response action. The rest has the same meaning as the corresponding part of the Signal.

- Time limit
The time limit represents the upper limit of time from receiving the signal to generating the response action. It consists of the time limit keyword "time limit" and the time interval with the unit is milliseconds.

Example of module response time is as follows:
Response: module A [Comp A, Comp B] signal: Comp C pub Topic θ; action: Comp A pub Topic β time limits 300.

The enumerated module response time condition acts on module A, which is composed of component A and component B. The signal is the data with topic θ published from component C. Module response action is the data with topic β published from component A with a time limit of 300 milliseconds.

2.1.4. Resource constraints
Resource constraints represent resource utilization constraints that components need to meet during their operation. Its structure is as follows: < Resource constraint condition declaration keyword, Component, Resource type, Resource constraint >. Resource constraint condition declaration keyword is "Resource". Component refers to the component that needs to meet the resource restriction condition. Resource type includes CPU and memory, and the Resource limit is the upper limit of using resources. CPU is the percentage, and memory is the specific value in terms of KB.

Examples of resource constraints: Resource: Comp A CPU 20 Mem 1024, the resource constraint acts on component A, with a CPU resource limit of 20% and a memory resource limit of 1024KB.

2.2. Automated Testing Method for Publish-Subscribe Distributed System
Automated testing method for publish-subscribe distributed system is based on the system status monitoring data provided by publish-subscribe middleware. In the test, the testing conditions described in Section 2.1 have been defined firstly. During the system operation, the running status information of components has been collected through publish-subscribe middleware, and whether the system works properly or not has been finally determined through judging whether the test conditions are satisfied or not. The method schematic is shown in Figure 1:
In general, the publish-subscribe middleware will monitor the status information in real time through the monitoring mechanism in the process of system operation. Monitoring information mainly includes the following types:

- Node status information, including: Node IP address, node host name, CPU occupancy, memory occupancy, list of components for topic publication and list of components for topic subscription.
- Application component status information, including: IP address of the node where the component is located, the host name of the node, the component name, the CPU utilization and memory utilization of the corresponding process of the component.
- Topic information, including: IP address of the node where the topic-related component is located, the host name of the node, the component name, domain name, topic name, topic type, topic status (publish/subscribe/unpublish/unsubscribe) of the topic, the data structure corresponding to the topic and the statistical information of publication and subscription of data.
- Data status information, including: Data action (send/receive), component name of the data, domain ID, IP address of the node of component of publication and subscription of data, topic name, time stamp, serial number.

The judgment of action consistency condition and module response time condition is based on the data status information obtained from monitoring. The sending and receiving actions of all components have been recorded by data status information collected by publish-subscribe middleware. The data has been filtered according to the sequence, time interval and other information defined by testing conditions to determine whether they meet the requirements or not.

Figure 1 Automated Test Flow Diagram
The node status information and application component status information have been used to judge resource constraints, and peak resource usage for target components has been recorded by publish-subscribe middleware. At the end of the test, it has been judged whether the peak value meets the requirements of the restrictive conditions or not.

The judgment of data legitimacy conditions is different from the monitoring data depending on the above conditions. It is necessary to obtain the actual data published by users. According to the requirement of data validity condition, the subscriber corresponding to the data has been created by publish-subscribe middleware, and the data published by the publisher has been subscribed. Because of the loosely-coupled nature of publish-subscribe middleware, there is little impact of new subscribers on the publishers in the components under test, which can ensure the accuracy of the testing results.

3. Automated Testing Tool for Publish-subscribe Distributed System

Based on the proposed automated testing method for publish-subscribe distributed system, an automated testing tool has been implemented on the premise of self-developed and DDS-compliant publish-subscribe middleware BlueDCS. BlueDCS can run on Windows, VxWorks, Roads, Linux and other operating systems. The automated testing tool can also test distributed systems based on BlueDCS running on the above heterogeneous platforms. According to its functions, the automated test tool is mainly composed of modules such as test condition compilation, test condition parsing, monitoring data collection and test result judgment, etc. The software deployment is shown in Figure 2:

![Figure 2 Automated Test Tool Software Deployment Diagram](image)

The testing condition editing module can provide a graphical component testing condition editing interface for testers. The incoming testing conditions have been parsed by the test condition parsing module, generating a logical structure to determine whether the component passes the test or not, and obtaining the test requirements for monitoring data. According to the requirement of the monitoring data proposed by the test condition parsing module, the monitoring data has been by the testing result judgment module, and whether the operation of the system under test meets the test conditions has been judged based on the logic structure generated before.
The interface of the testing condition editing module is the main interface of the tool, which is divided into the following two parts: Testing scenario construction and testing condition editing. The tester can construct a global view of the system under test through the relationship of primitives thanks to testing scenario construction interface. Testing scenario construction interface is shown in Figure 3.

![Testing scenario construction interface](image)

The box represents the components in the system under test, and the connection wire indicates that there is a relationship of publication of subscription of data between the components.

The tester can easily define and adjust test conditions thanks to testing condition editing interface. Testing condition editing interface is shown in Figure 4.

![Testing Condition Editing Interface](image)

The information that may be associated with the test condition in the system under test has been organized through the testing condition editing interface, and the tester can complete the editing of the test condition by simply selecting and entering.

The global view of the system under test and the component testing conditions form a component test case, and the testing tool can load different test cases and run them separately, as shown in Figure 5.
After the testing tool is loaded with the test case file, the tester presses the Start button to start the test, and the test tool will start the monitoring data collection module. The BlueDCS monitoring information of each node in the system can be collected by the monitoring data collection module by means of the monitoring mechanism in BlueDCS middleware then the tester carries on the test operation according to the test needs. After the test operation is completed, press the Finish button, and the testing tool can automatically judges the test results according to the monitoring data and test conditions and feed back to the tester.

It has been verified by practical application that a distributed system based on BlueDCS, with a scale of 10 nodes and each node deploys 50 components on average, can be effectively tested and its faults can be accurately located by means of the distributed system automated testing tool implemented in this paper in the integration and debugging phase of the system, which will greatly improve the efficiency of system development.

4. Conclusion
An automated testing tool has been proposed according to the problems of low test efficiency and difficult fault location in publish-subscribe distributed system test in the paper, and an automated testing tool has been implemented based on BlueDCS publish-subscribe middleware system developed independently. It has been verified by practical application that the practical application proves that the automated testing tool can efficiently complete the testing task and locate the fault in a large scale system. The next step is to optimize the monitoring data collection mechanism to meet the needs of large-scale system testing, reduce the overhead of monitoring data collection and storage, and further improve the adaptability of tools.

References
[1] OBJECT MANAGEMENT GROUP. (2009) Distribution Service for Real-time Systems Version 1.2.
[2] OBJECT MANAGEMENT GROUP. (2014) Real-time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol v2.1.
[3] Real-Time Innovations, Inc. (2013) RTI Connext DDS Core Libraries and Utilities User's Manual.
[4] Qu Zhijian, Lin Hongping, Jiang Junjun. (2015) Real-time Publishing And Subscribing Of Monitoring Information Flow Based On Newsboy Revenue Mechanism, Power System Technology, 39(9): 2665-2669.
[5] George Coulouris. (2008) Concept and Design of Distributed System. China Machine Press.
[6] Gerard Tel. (2004) Introduction to Distributed Algorithms, Second Edition. China Machine Press.
[7] Hou Xiuhong, Gao Fangfang. (2014) Survey of Content-based Publication and Subscription Systems, Computer Development & Application, (10): 10-13.

[8] Liao Chuang, Zheng Gang, Gao Qian. (2013) Research on Data Distribution Service of Ship Information System, Computer Engineering, (9): 94-97.