Application of UniTESK Technology for Functional Testing of Infrastructural Grid Software

Sergey Smolov

ISP RAS

ssedai@ispras.ru

Abstract

In this article some questions of testing of infrastructural Grid software on the standard compliance are discussed. Nowadays Grid-systems are one of the first-priority areas in computer science. The primary task is an effective usage of their advantages that is inseparably connected with the problem of software portability in Grid-systems. One of the simplest solutions of such problem is a development of standards on bundled software. Particularly, the compliance to the standard occurs as one of basic requirements to a system that is why its accomplishment should be checked with a high validity. In this letter the development of test patterns for infrastructural Grid software by means of UniTESK technology is considered. The result of program packet Globus Toolkit 4.2 testing upon compliance with WSRF 1.2 basic standard is given.

1. Introduction

The term "Grid" appeared at the beginning of 1990th in "The Grid: Blueprint for a new computing infrastructure" [1] collection under the editorship of Ian Foster as a metaphor of such an ease of access to computational resources as to power grid. According to Ian Foster, Grid-system (hereinafter referred to as GS) is a system, that:

a) coordinates the usage of resources in the absence of a centralized management;

b) supports standard, open and universal protocols and interfaces;

c) non-trivially supplies with a high-quality services.

Following by this definition, GS is a universal infrastructure of processing and storing distributed data, and of different services, called Grid-services, that are functioning in it. Different GS must support standard protocols and interfaces, despite of possible differences in architecture or specialties of realization. The purpose of the Grid standardization is to guarantee a portability of applications between different Grids, including systems that are built upon different infrastructural program packages.

1.1 Grid standardization

Realizations of GS had been appearing since 1995, when infrastructural program package Globus Toolkit appeared; nowadays it is de facto a standard of GS. It was made by Globus Alliance – the major international consortium in the area of Grid. By 1997 a European project of creating a program package for GS had begun, and it has brought to the infrastructural UNICORE software. By 2004 under the aegis of EGEE (Enabling Grids for E-sciencE) project, the gLite package had been released. In the connection with the great number of incompatible realizations of infrastructural Grid software, the necessity of unification and standardization became actual and active work on Grid standards creation started. Three groups of standards of the infrastructural Grid software are existing nowadays: the WSRF (Web Services Resource Framework), the OGSA (Open Grid Services Architecture) and the WS-Management.

1.2 Questions of Grid realizations testing

One of the specialties of the application domain is an existing of the incompatible, generally speaking, standards and some independent realizations. It is clear, that testing of the realization upon the standard compliance is a very actual problem.

It is worth to notice, that in the application domain the specificity of interaction between user applications and GS means occurs – in this case they are Grid- and Web-services and remote procedure calls.

The most widespread approaches to the GS testing are:

a) unit testing – a testing of different software modules. Particularly, Globus Toolkit developers use

JUnit for their realization checking.

b) integration testing – a testing of applications execution upon the infrastructural Grid software assembly. Typical examples of integration test scenarios are data bulk transfers and routine calculation implementations.

Both approaches are turned to realization errors detection. But they have a considerable defect in the context of compliance testing: there is no connection between tests and standards requirements. That is, it is impossible to draw a conclusion about compliance or mismatching to the standard by test results.

1.3 Technology UniTESK of the automated testing

Since 1994 the UniTESK technology of the automated testing has been developed. It was successfully used for testing of the different classes of program systems – program interfaces, telecommunication protocols and hardware. An access to the infrastructural Grid software is realized by different mechanisms of remote procedure calls and official protocols that are very close to the domain of applicability of the UniTESK, that's why this technology was chosen as a technology platform for tests development.

The test development with the application of the UniTESK technology accomplishes in the next seven stages:

1) requirement analysis and its formalization, formal specifications building;

2) requirements to the quality of testing formulation;

3) test scenarios, that are realizing such coverage, development;

4) test scenarios binding to the concrete target system by mediator development;

5) tests translation and compilation;

6) tests debugging and execution;

7) testing results analysis.

The important advantage of the UniTESK technology is an estimation of quality of testing possibility by calculating the requirements coverage. That is also the essential distinction from overwhelming majority of test patterns for GS. The calculation of requirements coverage and the constructing of oracles (special components that are checking the compliance of the target system to the specification) are automated.

Thus, the UniTESK technology allows driving widespread testing of a big class of program systems and, particularly, to develop test patterns for compliance problem solution.

1.4 OGSA and WSRF standards. Applicable domain of problem restriction.

The OGSA standard describes infrastructural middleware OGSI (Open Grid Services Infrastructure) between user applications and computational resources. It means that applications have no any possibility to interact with resources directly, but only by using infrastructural software. At the root of OGSI architecture the Grid-service concept lies that represents a mechanism of remote calls and was developed specially for Globus Toolkit 3.

At the same time with OGSA standard development, in 2004 the OASIS consortium suggested the WSRF standard. It also describes some infrastructural software, but based on not Grid- but Web-services. Correspondingly, the application domain of this research is exactly infrastructural software that is based on Web-services, because on this concept the most widespread infrastructural Grid software realizations are based.

In this research a possibility of UniTESK technology application for the functional testing of Grid software analyses, including the testing of standard compliance. Particularly, the next problems are considered:

1) representation of requirements to the services of infrastructural Gird software in formal UniTESK specifications;

2) development of mediators for impacts upon infrastructural Grid software;

3) development of testing scenarios for infrastructural Grid software.

2. Requirements to the realizations of infrastructural Grid software formalization

2.1 Regulating documents and requirements

to the realizations of infrastructural Grid software

As it is mentioned above, there are two standards that are used in GS development nowadays: OGSA and WSRF. The OGSA standard is a description standard, it does not contain any functional requirements, existing requirements are uncertainly expressed, descriptions of message formats and remote accessing protocols are not given. Therefore, the OGSA standard does not suit to be the base for the formal specification and based on it test pattern development.

The WSRF standard for standard formalization suites more. It contains 5 specifications:

1) WS-BaseFaults – determines format of error messages and the mechanism of their processing;

2) WS-Resource – determines WS-Resource concept itself, formats of messages and the semantics of management services;

3) WS-ResourceLifetime – determines mechanisms of destroying the WS-Resource;

4) WS-ResourceProperties – determines, in what way a WS-Resource is connected with an interface, that describes Web-service, and also represents the mechanisms of getting, changing and deleting properties of WS-Resource;

5) WS-ServiceGroup – determines an interface to the set of heterogeneous Web-services.

The Resource is a logical entity that has the following characteristics: identifiability, lifetime and set of zero or more properties, which are expressible in XML Infoset. The WS-Resource is a composition of the Resource and the Web-service, by using its methods or fields an access to the Resource is accomplished. The WSRF standard is very convenient to analyse because of its structuredness. For example, the bigger part of functional requirements are supplied with RFC 2119 keywords – MUST, SHOULD, MAY. Moreover, most requirements are supplied with blocks of descriptions and examples that have been written on pseudocode that looks like WSDL 2.0 Web-services description language.

Certain parts of standard have different levels of obligatory in RFC 2119 gradation. That is, a WS-Resource must realize requirements of WS-Resource and WS-ResourceProperties specifications, also it should realize requirements of WS-Base-Faults specification and it may satisfy requirements of WS-ResourceLifetime. Consequently, under the test pattern development, the first two specifications should be taken into account first of all. Such approach considerably simplifies the test pattern structure and reduces it, but retains tests correctness as solution of the problem of compliance.

There are following message exchanges in WS-ResourceProperties specification:

1) GetResourcePropertyDocument – getting all the properties of the WS-Resource;

2) GetResourceProperty – getting the certain property of the WS-Resource;

3) GetMultipleResourceProperties – getting multiple properties of the WS-Resource;

4) QueryResourceProperties – determining the structure of WS-Resource properties and querying the requests upon them;

5) PutResourcePropertyDocument – changing of the “old” Resource Property Document (a set of all properties of the WS-Resource) by the “new” one;

6) SetResourceProperties – changing some properties of the WS-Resource (i.e. a composition of the three following message exchanges);

7) InsertResourceProperties – adding new properties;

8) UpdateResourceProperties – changing values of the existing properties of the WS-Resource;

9) DeleteResourceProperties – deleting some properties of the WS-Resource.

In the course of the analysis of standard 325 functional requirements had been marked out, 29 – in WS-BaseFaults specification, 12 – in WS-Resource, 51 – in WS-ResourceLifetime, 159 – in Ws-ResourceProperties and 73 – in WS-ServiceGroup.

2.2 Formal specification development

Every message exchange in WSRF corresponds to pair <request - response>, where in the capacity of the response can be message with the returned value or error message. Thereby WSRF message exchanges can be modeled as function calls with the returned values, error messages can be modeled as exceptions.

Due to in the concerned method of the requirements to the infrastructural Grid software formalization, the Web-service is modeled as an object of some class, and message exchanges between client and realization are represented as calls of this class methods.

Altogether a half of all WSRF requirements was formalized (nearly 160 requirements) and about 60% requirements from WS-Resource, WS-

ResourceLifetime and WS-ResourceProperties specifications. The whole set of requirements can be separated into two basic groups: syntactical and functional requirements. Syntactical requirements impose constraints on a structure of messages and relationships between the fields of one message. Functional requirements correspond as restrictions upon the functionality of requests processing and connections between a content of request and a content of response. It is recommended to check syntactical requirements in mediators at the stage of message parsing.

2.3 Mediator development

The main function of mediator, as a component of a test pattern, is an establishment of correspondence between a model object (object of specification class) and a target system. In case of infrastructural Grid software testing, there is no possibility to have an access to fields and methods of system, that is why it could be accomplished only by sending appropriate requests and receiving and parsing responses. In concerned method mediator transforms parameters of a specification method into SOAP/HTTP message and sends it to the target system on established TCP connection. Received responses are checked by the mediator, the correspondence to syntactical requirements and parsed by mediator too. Then mediator forms the returned value of the specification function.

The peculiarity of the mediator development is that existing realizations (particularly, Globus Toolkit 4.2) do not satisfy to syntactical requirements of the standard. For the testing implementation the adoption of mediators by standard violations was needed. Also by analyzing the source code of infrastructural Grid software Globus Toolkit 4.2 and taking some experiments with realization was established that realization does not support the following message exchanges – PutResourceProperties and SetResourceProperties.

2.4 Test scenarios development

Under test scenario for testing of the infrastructural Grid software of compliance to the WSRF standard development, in the capacity of an automate state identifier it is recommended to use the cardinality of Resource Property Document of the WS-Resource, i.e. an amount of WS-Resource properties. On the one hand, such feature considerably reduces the complexity of supposed state graph, and, appropriately, a time of tests execution. On the other hand, under such definition of state of the automate the determinacy of graph retains that is important for a correct operation of the UniTESK iterator.

Transitions between automate states are accomplished by offering stimuli into the target system. The role of stimuli in the case of development of the test pattern for infrastructural Grid software play mediator methods calls. As soon as synchronous model of the target system is used, for every specification method it is possible to create a separate scenario method, which will go over the parameters of method and will call (implicitly) mediator for testing impact and an oracle for checking the returned value on correctness.

3. Experience of practical testing of the infrastructural Grid software

The method, which was represented above, was used under development of the test pattern for checking the compliance of the infrastructural Grid software realizations to the WSRF standard.

3.1 Target system review

The object under testing in this research is a program packet Globus Toolkit 4.2. Globus Toolkit 4.2 uses protocols of standard Web-services and mechanisms of services description, detection, control, authentication and authorization. This program packet includes components that can be used for constructing containers. In these containers Web-services, which are written on Java, C and Python can be placed.

In accordance with Globus Alliance, the Java WS Core component of Globus Toolkit 4.2 (it supports the Web-services development and execution of Java applications) realizes requirements of WSRF standard. In this research the problem of compliance was decided exactly for this component.

3.2 Implementation testing

Test scenarios that were developed by using JavaTESK instrument include scenarios for eight methods: GetResourcePropertyDocument, GetResourceProperty, GetMultipleResourceProperties

Insert-, Update- and DeleteResourceProperties, and also ImmediateDestroy and SheduledDestroy.

It is worth explain, why these eight message exchanges (i.e. specification methods) were chosen. At the time of test pattern development it was supposed, that by test pattern using all services of the container of the Java WS Core component will be tested. By default, there is 34 Web-services in container, and only 23 of them support message exchanges, that are mentioned above and are contained in WS-ResourceProperties and WS-ResourceLifetime specifications of WSRF standard. Correspondingly, these 8 message exchanges is both simple enough to specification methods development (the QueryResourceProperties message exchange is not so simple for testing) and allow to run testing of the compliance to the WSRF 1.2 standard.

3.3 Testing results

In this research the realization of infrastructural Grid software Globus Toolkit was tested by the facilities of the UniTESK (JavaTESK) technology. In the capacity of testing results reports about requirements coverage, which were generated by the instrument, acted. Nearly 60% of system functionality was covered, wherein this test pattern does not yield to tests that are used by Globus Toolkit developers. They measures code coverage by tests for quality of testing determination with the JUnit and Clover instruments. These instruments allowed developers to determine that their unit-tests had covered 60% of the functionality of system too (i.e. Java WS Core component). However, as it was mentioned above, these tests cannot report about the compliance of Realization Globus Toolkit 4.2 to the some standard. Thus, existing and developed tests not only complement each other, but allow considering the realization from the different points of view.

Under the realization testing some semantic discrepancies to the WSRF 1.2 standard (in InsertResourceProperties and UpdateResourceProperties message exchanges) were revealed. Particularly, these methods allow to add and change values of properties of the WS-Resource with different identifiers (which are called QNames) that is forbidden by the standard requirements.

4. Existing methods and approaching of Grid infrastructural software testing

In 2006 ETSI (European Telecommunications Standards Institute) organized an expert group for development a test pattern for Grid compliance testing [2]. Method that is being developed in ETSI is based on the development on large amount of test cases on TTCN-3 programming language. The prototype of such test pattern on TTCN-3 language is represented in [3]. Test pattern is not connected with any standard of infrastructural Grid software. Instead of standard requirements checking this test pattern checks applicability of Grid in typical use cases – statement of the computational task in query, task execution on one of the computing nodes, result delivery.

In [4] authors propose an approach to the Grid testing that is close to the approach presented in this article. Authors offer to use a formalism of abstract state machines (ASM) and automatically generate test sequences from automate bypass. Questions of Grid standards analysis and requirements formalization and formal model building are not considering.

5. Conclusion

In this research the problem of testing of the compliance of the infrastructural Grid software realization Globus Toolkit 4.2 to the standard WSRF 1.2 was being solved. This standard was analyzed and a catalogue of its requirements was created. Interfaces and the structure of Java WS Core component of the Globus Toolkit 4.2 were explored also. On basis of these data the test pattern for this program packet was developed by using UniTESK (JavaTESK) technology and testing was carried out. The testing has showed that the realization Globus Toolkit 4.2 complies with the WSRF standard and has revealed a lack of some unnecessary requirements accomplishment, both functional and syntactical. Also testing has showed that the UniTESK technology is applicable for testing the infrastructural Grid software, particularly, there are the following peculiarities of its application:

1) message exchanges with Web-services are essentially modeled by specification functions;

2) for test pattern development a class library for automatization of building different (and “incorrect” also) messages of Web-services is necessary.

In the capacity of directions for the future research we consider a development of test pattern, that is checking a specific requirements to the services of the infrastructural Grid software, like a service of bulk data transferring, service of creation and management of computational resources and so on.

6. References

[1] Ian Foster *The Grid: Blueprint for a New Computing Infrastructure.* — Morgan Kaufmann Publishers. — ISBN 1-55860-475-8

[2] S. Schulze. Achieving Grid Interoperability: The ETSI Approach. *The 20th Open Grid Forum - OGF20/EGEE 2nd User Forum.* Manchester, UK. May 7 - 11, 2007

[3] T.Rings, H.Neukirchen, J.Grabowski. Testing Grid ApplicationWorkflows Using TTCN-3. First International Conference on Software Testing, Verification, and Validation, ICST 2008, Lillehammer, Norway, April 9-11, 2008.

[4] Lamch, D.; Wyrzykowski, R. Specification, Analysis and Testing of Grid Environments Using Abstract State Machines. *International Symposium on Parallel Computing in Electrical Engineering, 2006. PAR ELEC 2006.* 13-17 Sept. 2006 Pages:116 - 120