A FRAMEWORK FOR TESTING TELECOMMUNICATION SERVICES

G.S. Vermeer, M.F. Witteman, J. Kroon.

KPN Research

P.O.Box 421

2260 AK Leidschendam

Netherlands

Email:

G.S.Vermeer@research.ptt.nl

M.F.Witteman@research.ptt.nl

J.Kroon@research.ptt.nl

ABSTRACT

Service testing is a new development pushed by the rapid increase of the number of services offered by telecom operators. This paper defines a framework for the test process related to the operation of new services. First an overview is presented of terms and concepts in the service life cycle. Then a procedure is defined that covers all testing activities that should be considered before and during service operation.

Keywords

Service testing

1. INTRODUCTION

In the turbulent telecommunications world, the rapid introduction of new telecommunication services can supply Service Providers with a competitive edge. The introduction of an Intelligent Network architecture will enable operators to shorten the introduction of new services from many years down to a number of months, or even to a couple of weeks. Furthermore operators will have the possibility to create new services in their own Service Creation Environments.

In telecommunications, on the other hand, service providers can hardly afford failures of services. Failures of the network may have a catastrophic effect on the operators' image as

a reliable service provider. This threat provides the need for 'Service Testing'. Before a telecommunication service is deployed into the network, it should be ascertained that the service is functionally correct, and that the service does not harm the already installed base.

This paper aims at providing a framework for the test activities considering validation of new telecommunication services. Rather than presenting new and ambitious test techniques we will investigate the role of testing in the service life cycle. This framework is largely based on the Intelligent Network architecture. The framework however is fit for application on other architectures as well.

Section two will explain the various terms and concepts related to the service life cycle. Section three will introduce a procedure for all testing activities that should be considered before, and during service operation. Section four will conclude the paper.

2. SERVICE LIFE CYCLE

This section summarises some terms and concepts that are used in the world of service creation. Understanding these concepts is important in order to see the relation with the testing activities presented in the next section. Readers that are very familiar with service design may wish to continue with the next section immediately.

2.1. What is a service?

A very basic aspect is the notion of what a service actually is. It appears that many different definitions are used over the various documentation. This is due to the fact that services can be seen from commercial, technical, and user perspectives. As testing is merely focusing on the technical perspective a technical definition would probably be most useful.

For this reason we introduce the following definition of a service:

A (telecommunication) service is a stand-alone commercial offering, which can be identified by all parts of hardware and software which are especially designed, installed or configured, to enable the service.

We will use this definition of a service for the service testing activities suggested in this paper.

An example may clarify the meaning of this definition. Consider a service, commercially defined as: "A time dependent call forwarding service that connects all calls to B from 17.00 till 9.00 to another number C". The software that enables this service comprises code that checks the B-number, and decides upon its re-routing. The required hardware could consist of the clock mechanism that provides the appropriate time signals for the service.

2.2. Stages in the service testing life cycle

In order to understand the various aspects that contribute to the realisation of a service it is useful to observe the stages that are traversed during the service development.

The Eurescom P103 (Eurescom, 1992) project has defined a life cycle that we consider quite appropriate (see Figure 1).

In this paper we like to position most of the service testing activities after the service implementations tasks are completed. The service testing tasks take place before installing the service into the network. The result of service testing therefore should be a service which exhibits correct behaviour and is ready for being installed in the operational network.

A minor part of the service testing activities however will take place during service operation, which comprise the phases after the service installation, and before service de-installation. We will come back to this anomaly in section three.

Figure 1 Service Testing in the Service Life Cycle

3. THE INTELLIGENT NETWORK ARCHITECTURE

Services can be provided using the 'Intelligent Network' architecture. In Figure 2 a possible configuration of an intelligent network is pictured. In this architecture an IN-(service) call is detected at the Service Switching Point (SSP). The SSP will consult the Service Control Point (SCP) for instructions. The SCP contains the service logic and service data that are used to provide IN services. The SCP can access data in a Service Data Point (SDP) either directly, or through a signalling network.

Figure 2 Intelligent network architecture

The intelligent Peripheral (IP) provides specialised resources for customisation of services, and supports flexible interaction between a user and the network.

An IP can be used, for example, to play announcements like 'type your PIN' in a credit card validation service. The Service Management Point performs service management control, service provision control, and service deployment control. The Service Creation Environment (SCE) is used to define, develop and test an IN service and input it into the SMP.

4. SERVICE TESTING

At a first glance, the testing activities that should be performed with the introduction of a new service should be quite limited. One could suggest that a simple check on the availability of the functionality would satisfy. There are however a number of reasons to apply a careful installation procedure:

- Realisation of a new service may require modification to the service platform, or it may affect the platform functionality unintentionally;
- Realisation of a new service may require modification to existing services, or it may change the behaviour of the existing services unintentionally;
- Realisation of a new service may seem to be successful but a malfunction may show up (long) after installation, due to complex relations with the various network elements.

To assure the proper working of service the following 5 tasks should be performed:

1. Test the service platform
2. Test the 'new' service
3. Perform regression tests
4. Perform Service Interaction tests
5. Perform Service Management

Before installing a service into a IN structured network, the network itself has to be tested; This first task is in this paper referred as platform testing. The second step considers the new service. It has to be tested whether the functionality of the new service conforms to its specification. When a new service is installed and activated, it has to be tested (task 3) whether the already installed services are still operational. Then, before making the service operational, task 4 checks for unwanted service interactions.

When the service is operational still tests are to be performed during service management (task 5). These tests consider aspects like 'is the service still working?', 'is the performance still satisfactory', etc. In the following subsections the test steps will be discussed into more detail.

4.1. Platform Testing

Before installing new services into a service platform, the platform itself has to be tested. In the IN-standards (CCITT,1992), a service is composed of so-called Service Independent Building Blocks (SIBs).

It can be said that these building blocks are part of the platform and should be subject of platform testing. It is possible to test a SIB in some artificial environment (like simulation). However, to test a single SIB implemented in a SCP will be difficult. For this purpose a 'test service' may be useful. Such a service exhibits not necessarily any functionality employable by end-users, but supports testing the relevant part of the service logic.

The communication between the network elements SSP and SCP (and in some cases SDP) will make use of the standardised core-INAP protocol. For testing this protocol stack, conformance test standards are (being) developed within ETSI. Before deploying these network elements into the network, these elements should have passed these conformance tests.

Besides the functional behaviour, non functional aspects like performance should be tested as well. It should be verified that the platform exhibits sufficient performance to run the desired services. This can be done by either performance calculations (including prediction algorithms) or by performance benchmark tests.

4.2. Testing the new service

Once the service is created the new service will be tested in a test environment. Naturally, this environment should be representative for the operational environment. Of course measures should be taken to ensure that errors which occur have no effect on the operational environment.

Several approaches can be taken to test whether the service complies to the specification, i.e. the service subscriber's needs. These approaches differ in costs, ease-of-use, and fault detection capabilities. A straight-forward technique applies simple end-to-end testing of the service behaviour. A more advanced approach provides in monitoring network or protocol traffic on a number of strategic locations. The most complex and exhaustive testing can be performed by stand-alone testing of a single network element. Below these approaches are discussed in detail.

4.3. The end-to-end service testing approach

Figure 3 pictures the most basic approach towards the testing of a new service.

Figure 3 End-to-end service testing approach

In this approach the service is downloaded into the (test)platform. The tests are controlled at the user-end of the platform, i.e. by simulating the service user behaviour using DTMF (terminal key pad buttons) access. The tests are observed at this interface as well. This type of testing can be referred to as 'black-box' testing.

For example, testing a Call-Forwarding service can be done using three phones. Suppose phone B is forwarded to phone C. Subsequently phone A dials B's number. Phone C rings and when picked up a speech channel is established between A and C. The advantage of this approach will be the low costs and short time needed to set up the test configuration. The description of the tests could even be done using natural language. In that case the specifications would be easy to write.

The major drawback of such an approach is the limited coverage of the tests. Applying the terminal access as a PCO implies that part of the service logic that deals with exception situations cannot easily be tested. This is caused because the terminal access is not, or hardly, able to provide in network failures or resource limitations. Another drawback of this approach is the lack of fault localisation. If errors occur it will be very difficult to determine which part of the network caused the error.

4.4. The monitored end-to-end service testing approach

One of the drawbacks of the approach described above are the possibilities of fault localisation. A more refined approach is the so called 'Monitored end-to-end testing' (Figure 4).

Figure 4: Monitored end-to-end testing approach

In this approach, again the tests are controlled from a user point of view. To observe the network's 'internal behaviour' monitors can be installed at several places. Obvious observation points for monitor equipment are the communication links. These links are often based on international standards (core-INAP). These monitors are (or will be) therefore commercially available. In some cases network elements will provide possibilities to monitor the inside behaviour of the network elements as well.

For the test description two approaches can be used. One approach is to use the same kind of test specification as used by the 'end-to-end' test approach. In this case the monitors are only used when errors occur in order to locate the error sources.

Another approach can be found in (Gabrielli, 1992). In this approach all expected observations visible at the monitors are fully described. In the test specification not only the external behaviour (like in the end-to-end testing approach) will be described, a description of the internal events in the SSP and SCP, the INAP messages exchanged over the SS7 interface, and the queries in the SDP are to be specified as well. On the one hand this provides detailed information on what is tested. On the other hand the test specification will tend to become very large and difficult to read.

Looking at the advantages/disadvantages, using the monitored end-to-end testing approach, information is gained about the IN internal events. This provides the opportunity to locate the source of the error. The testing port (the terminal access) is still the same as described for the previous approach. Therefore the testing power will still be limited to the possibilities of the user interface. We will refer to this type of testing as 'grey-box' testing.

4.5. The Service Conformance Testing approach

The most powerful test method namely the Service Conformance Testing approach can be found in Figure 5.

Figure 5 Service Conformance Testing approach

Using this method the tests are mainly focused on one network element, in this configuration the SCP. The service specific software largely resides in the SCP. The part of the service in the SSP will in most cases only affect some parameter settings (like the update of trigger detection points)

In Figure 5, the PCO is located at the communication link, which exhibits the standardised core-INAP protocol. Tests are controlled by this access. Using the core-INAP interface provides extensive possibilities for testing, that go beyond the level of valid behaviour testing. The core-INAP interface provides possibilities to perform tests for invalid behaviour and data encodings as well.

This testing approach has a number of similarities with 'traditional protocol conformance testing' which is defined in ISO9646 (ISO,1994). It is therefore likely that the TTCN language will be suitable for this approach to Service Testing. The test specification for the call forwarding service will look like:

```

1! InitialDetectionPoint (Cf,B,C.....)
"forward phone B to phone C"
    !InitialDetectionPoint(A,B,...)
    "A dials B number"
        ?Connect (A,C,...)
        "Phone A is connected to phone C"
  
```

In this example the action of forwarding a phone B to phone C is translated in the INAP message 'InitialDetectionPoint(Cf,B,C,...)'. Subsequently phone A dials B's number (InitialDetectionPoint(A,B,...) and the SCP will provide the instructions to forward the call to phone C (Connect(A,C,...) As the PCO is now located inside the Intelligent Network, we will refer to this type of as 'white-box' testing.

The advantage of such an approach lies in the extensive testing power. This testing power provides possibilities to perform a number of robustness tests which are not possible with the approaches described before. However, using such an approach will require still a number of end-to-end tests to ascertain interoperability. The disadvantage of this an approach clearly lies in the effort needed to set up a test campaign and to produce the test

In protocol conformance testing the ! stands for 'Send' while the ? stands for 'receive'

suites. In a 'time-critical' provision schedule the large effort may be a bottle-neck in the testing process.

4.6. Which approach?

The choice of which approach should be used, depends on a number of factors. On one hand we have to deal with factors like 'required quality of service', c.q. required level of confidence'.

On the other hand we are limited by the available resources and time. At this moment there is little core-INAP conformance test equipment, and no complete test specifications available. Furthermore the lack of experience in service conformance testing may cause unacceptable delays in the tight testing schedule.

It seems therefore reasonable to start gaining experience in the field of service testing by applying manual end-to-end tests. Along the way of practice the process can be improved by automation and the use of more powerful methods. The generation and execution of tests could be automated, while the test methods could shift from end-to-end testing towards service conformance testing.

4.7. Regression tests

Even when a newly installed service is not activated yet, it may interfere with the functionality of other services; the addition of the new service to the installed base may have caused changes to the existing services. Therefore it is necessary to perform tests on existing services as well. These type of tests are called regression tests.

In Figure 6 a scenario for service regression tests is pictured. Before starting the regression tests, the new service should be completely tested using one of the approaches discussed in the previous paragraphs. To assure the other services are still working correct, regression tests have to be executed, by performing a subset of the functional tests for each services.

Figure 6 Service Regression tests

After the execution of all regression tests the standard set of regression tests can be extended by a subset of the functional tests for the new service. (In Figure 6, the service D.)

From the Figure 6, it can be seen that regression testing may involve a serious management problem. Like pictured, the set of standard regression tests will grow with the introduction of each new service.

For reducing the regression test set two alternative approaches can be used. Instead of using a subset of the functional tests, one can figure that it might be possible to use some 'test service' for the regression test. This service should contain features of all already installed services. Whether such an approach is feasible is hard to say.

Another possibility might be the use of categories and priorities for the selection of a regression set. This principle requires all functional tests to be categorised (e.g. according to the involved SIBS), and prioritised (e.g. according to the impact of the failure). Although the complete set of available tests grows with the addition of new services, it would be possible to select a constant number of regression tests based on categories and priorities.

4.8. Interaction tests

In the previous paragraphs it was stated that after testing the service on functionality, regression tests are to be performed to verify that the already installed services are still working correct. Therefore a subset of the functional tests of each installed service is executed. This is however not enough for deploying the service into the network. When each individual service is working correct, this gives no guarantee yet that the services operating concurrently will work correctly together. Services may conflict with each other and even with (other instances of) them-selves.

Two types of service interaction can be identified. The first type is inherent to the service logic and can (should) be detected in the specification. Smart algorithms can detect these service interactions, and alteration to the service can prevent the interactions to occur. The second type however, is depending on dynamic service data values, and is difficult to recognise at specification time.

In practice however it is yet not possible to detect undesirable service interactions in the specifications. Therefore some kind of 'service interaction' test must be executed to prevent the network from service interactions. These tests can be performed either in a simulation environment or in a real test environment.

An interesting approach can be found in (Razol, 1994). A simultaneous execution of functional tests can detect the second kind of interactions. When all functional tests executed simultaneously pass, it can be argued the network is free of service interactions.

4.9. Service management tests

Once the service is operational, test activities should still be performed. These test activities fall into the category of fault management. Within the Telecommunication Management Network (TMN) (CCITT, 1992), fault management is defined. Fault (or maintenance) management consists of a set of functions which enables the detection, isolation and correction of abnormal operation of the telecommunication network and its environment.

4.10. TMN Fault Management

Within the TMN standard the fault management is divided into the following four types of actions:

- Alarm surveillance (some Network Element (NE) or link is not working)
- Fault Localisation (where did the fault occur?)
- Fault Correction (resolve the fault)
- Testing (is a service still working)

Because this paper aims at service testing, the first three types will not be discussed. These types of actions consider errors, which will cause notifications. This error is subsequently localised and corrected.

In this report we concentrate on the testing type of fault management. Within M.3400 (CCITT, 1994) two types of testing activities are identified. Either the TMN requests the Network Element to test a service, and send back the report, or the TMN requests access to a Network Element and performs some testing itself. These types of testing can be used either preventive (testing if a service still is operational) or for fault location (after complaints about the service are received). Although Service Management is very important in keeping the service alive, unfortunately this area is still quite unexplored.

5. ACKNOWLEDGEMENTS

We like to thank the people within Eurescom, KPN Research and PTT Telecom who have contributed to the ideas that were presented in their papers. We owe our colleague Wilbert Schelvis for carefully reviewing this article.

6. CONCLUSIONS

This paper presented a framework for Service Testing. Before testing the new service itself, it should be assured that the platform is operating correctly. The new service should be tested in three steps. Within this frame several possibilities are available to ascertain that both the new service and the installed base of services, are operating correctly. Finally, when the service is in operation tests can be performed to assure that the services are still working and to locate error sources in case of failures.

Although the area of service testing is relatively new, a number of already available testing techniques can be reused for service testing. Already for basic telephony services testing call generators and ordinary telephone equipment is used. These equipment can be reused for service testing as well.

Also concepts and methodology from the conformance testing world can be applied for service testing. The really new aspect on service testing will be the time available for testing. To meet market goals the pressure for a rapid service introduction will grow. Therefore the time left for service testing will be under pressure more and more.

This time factor will become more and more important in the choice for a test methodology. It is therefore a challenge for the test research community to find time efficient solutions

7. REFERENCES

- Eurescom-P103 (1992) 'Evolution of the Intelligent Network, deliverable No.2'.
- H. Velthuisen et al. (1993), 'A feature Interaction Benchmark in IN and beyond', Bellcore Memorandum TM-SV-021982
- CCITT, Q1201 (1992), 'Principles of Intelligent Network Architecture'.
- ISO/IEC 9646: April 1994, Information Technology, OSI Conformance Testing Methodology and Framework.
- S.P. van de Burgt, J.Kroon, E.Kwast, H.J.Wilts (1990), The RNL Conformance Kit, Protocol Test Systems, IFIP.

- E. Anders, J. Ellsberger, A. Wiles (1994), Experiences with Computer Aided Test Generation, proceedings IWPTS conference 1994.
- L. Gabrielli, P. Marchese, (1992) 'Intelligent Network Service Testing', ICCS, Conference on IN, Tampa, USA.
- R.J. Heijink (1994), 'FAITH, a general purpose protocol test system for ISDN', Computer Networks and ISDN Systems (1994), 1581-1593.
- P.Razol et al (1994), 'Service Interaction and Test Generation', proceedings ICIN conference.
- CCITT, 'M.3010 (1992), principles for a telecommunications management network'.
- CCITT, 'M.3400 (1992), TMN Management Functions'.

8. BIOGRAPHY

Gert Vermeer graduated in 1990 at the department Informatics at the University of Twente. After his study he joined KPN Research, where he started working in the field of testing. He participated in several test projects for FTAM and other OSI protocols. Apart from test development he contributed to the methodology for interoperability testing. Currently he is responsible for the development of various IN test suites.

Marc Witteman graduated in 1989 at the department Electrotechnical Engineering at the University of Delft. In the same year he joined KPN Research, where he started working in the field of conformance testing. He participated in several test projects for GSM, IN and ATM protocols. One of his favorite subjects is the investigation of testability and the design of test architectures. Currently he is mainly involved in the validation of IN Services.

Jan Kroon studied Mathematics and Physics at the University of Utrecht in the Netherlands. He joined KPN Research in 1986. He has been a project leader of research projects in the field of conformance testing, with focus on automated test generation from formal specifications. In 1992 Jan Kroon worked for the European Telecommunications Standard Institute. His group produced a standard on the use of the specification techniques SDL. In June 1995 he completed a thesis with the subject "Specification and Testing of Telecommunication Systems" at the University of Bern.