A Multi-Service and Multi-Protocol Validation Platform - Experimentation Results

Ana Cavalli[1], Amel Mederreg[1], Fatiha Zaïdi[1], Pierre Combes[2], Wei Monin[3], Richard Castanet[4], Marcien MacKaya[4], Patrice Laurençot[5]

[1] Institut National des Télécommunications- CNRS Samovar
9 rue Charles Fourier 91011 Evry Cedex
{ana.cavalli, amel.mederreg, fatiha.zaidi}@int-evry.fr
[2]France Telecom R&D/DTL
38,40, rue du Général Leclerc, 92794, Issy les moulineaux, France
pierre.combes@rd.francetelecom.fr
[3]France Télécom R\&D, Technopole Anticipa
2, avenue Pierre Marzin, 22307, Lannion Cedex, France
wei.monin@rd.francetelecom.fr
[4]Université de Bordeaux 1 -Labri
351, cours de la libération 33405 Talence Cedex
{richard.castanet, marcien.mackaya}@labri.u-bordeaux.fr
[5]Université de Clermont-Ferrand- Limos
34, Avenue Carnot, 63000, Clermont-Ferrand, France
laurenco@isima.fr

Abstract. This article presents the implementation of a validation platform based on formal methods and the experimental results obtained. This platform allows performing conformance and interoperability tests, analysing the specification and constructing a performance model for the services. It covers all stages of the validation which are: formal specification, test architecture definition, test generation and execution for the defined architecture, and performance evaluation. The test methods and architectures used here make it easier to detect and localise errors. The platform has been constructed within the framework of the RNRT (National Telecommunications Research Network) platform project, PLATONIS. This platform is composed of a network integrating the different sites of the project partners. The principal application domains for the platform are telecommunication systems and mobile telephony. In particular, two different cases study are presented that illustrate the platform's applicability to the test of mobile 3rd generation protocols and services using WAP[1], GPRS[2] and UMTS[3]. Nevertheless, the platform is generic and can be used for other types of communication protocols and services.

1 Introduction

In the last few years major progress has been achieved in the area of networks and computers, particularly concerning Internet and mobile networks. This evolution has

[1] Wireless Application Protocol
[2] General Packet Radio Service
[3] Universal Mobile Telecommunication System

strengthened the idea of mobile telephony over Internet. This last has entailed the design of new protocols and services. However, the architectures implemented by using these protocols and services interconnect heterogeneous elements that needs to be tested in order to validate their interoperability. Tests and trials on real platforms is crucial for all the actors involved, such as, operators, service providers and equipment manufacturers.

This article presents the implementation and the experimentation results of the PLATONIS multi-protocol and multi-service validation and experimentation platform [9], [14]. The work has been approved and financed by the RNRT program and is the result of the collaboration between several research labs and industrial groups. The objective of PLATONIS platform is to help industry (operators, service providers and equipment manufacturers) and research to test the conformance, the interoperability and evaluate the performance of new communication protocols and services. It assures the reliability by detecting functional errors (output and transmission errors) and performance problems. The platform has been deployed over several sites: Evry (at INT), Bordeaux (at LABRI), Clemont Ferrand (at LIMOS) and Issy-les-Molineaux (at France Telecom R&D).

The implementation of the platform integrates the network configuration part and the open source WAP protocol stack called Kannel [12]. This implementation respects the standard established by the WAP Forum. The results obtained by applying the platform encompass all test phases: specification, test generation, test execution and, also, specification analysis. These results also include the definition of new test methods and architectures. Indeed, in the cases studied, it has been noted that an implementation under test is often embedded in a complex system that does not provide a directly accessible interface to the implementation under test. This is the case with the different WAP protocol layers. The fact that there is no direct access makes it impossible to rely only on classical test methods. This makes it necessary to define test architectures that incorporate Points of Control and Observation (PCO) and Points of Observation (PO) in the implementation and use appropriate test methods.

Concerning the formal specification of the protocols, the WAP protocols were described using formal methods in order to make it easier to apply automated methods for test generation. These specifications were done for the different layers involved and in particular for WSP[4], WTP[5] and a simplified version of WAE[6].

This article also presents the formal specification of new services for cellular phones that require Internet access for their provision. In the project, the services chosen were based on the location of the subscriber and ran with WAP over GSM, GPRS and UMTS infrastructure. The services were formally described, implemented and validated. The validation was done using test scenarios that were generated from the specifications (an approach based on an XML parser have been followed).

The generation of test scenarios was also done for the protocols. The execution of these scenarios was done on the protocol stack and using simulated and real terminal devices (cellular phone and PDA's -Personal Digital Assistants).

---

[4] Wireless Session Protocol

[5] Wireless Transport Protocol

[6] Wireless Application Environment

The article also presents the elements forming the approach that allows integrating the functional validation and the performance evaluation in the same environment. This approach was used on the same protocols and location services. Let us note that the performance evaluation of a system needs to be considered as soon as from the conception phase and stay coherent with the functional validation.
In the following sections the article first introduces the configuration of the platform and presents the new test architectures (section 2). In section 3, the formal specification of the WAP protocols and the location based services are presented. In section 4, the derived performance model is explained. In section 5, the experimental results obtained using the implemented services and protocol stack, are presented. Finally, in section 6 the conclusions and perspectives for the work are given.

2 Test Methodology and Architecture

2.1 Platform Configuration

The platform includes several sites corresponding to the partners involved in the project. Only the academic sites are open. The platform can be accessed through a mobile phone, a PDA or a simulated mobile terminal. In the case of the mobile phones or the PDA's, the access is authenticated using a RAS (Remote Access Service). This service allows the access to the Open Source Kannel WAP protocol stack. This stack connects the mobile terminals to the different HTTP servers available on the platform (Apache server, IIS 4.0) and also to the WAP service provider servers (see Fig. 1). A detailed description of the installation for each site is given in section 2.3.2.

2.2 Test Methodology

The objective is to define a methodology and the architectures for the validation and trial of the protocols and services. It is also a goal to cover and automate all phases of the test process: from the specification of the system under test to its execution on the real platform. This last allows performing conformance and interoperability tests. The interoperability tests serve to validate that different implementations interact correctly, that is, the implementations provide the global service as expected and conform to the standard. This type of tests allow verifying the interoperability between, for instance, an application running on the terminal and another on the server. On the other hand, conformance tests verify that the implementation under test follows the standard, as for instance, verifying that one of the WAP protocol layers functions correctly.

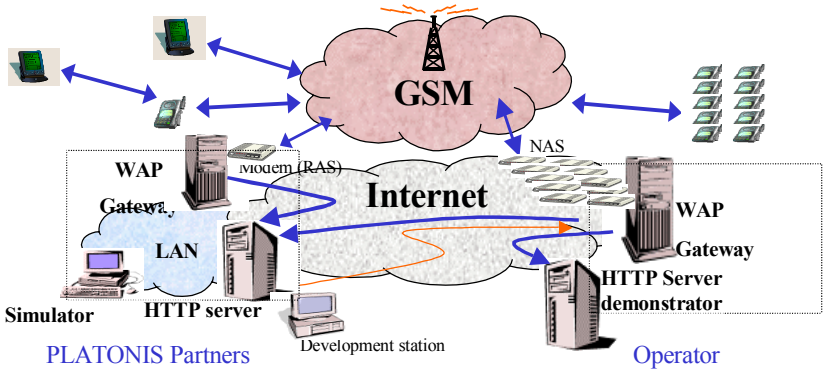

Fig. 1. The PLATONIS platform.

The test methodology used here is based on two main elements: generation methods and architectures. Below are described the different steps that constitute this methodology:

1. First, a precise representation of the system is provided. The description language used is SDL [4]. The specification needs to take into account the test architecture and the interactions with the environment.
2. Next, the tests that need to be made are selected by defining test objectives. It means choosing a strategy for the selection of tests using predefined criteria. To generate the tests, test generation algorithms are used that have been conceived for embedded testing and can be easily adapted to objectives based testing. [1] provides a detailed description of these algorithms.
3. The generated tests are then executed over the test architectures that are proposed in this article and that are described in detail in the following section.

2.3 Test Architectures

At first, the PLATONIS platform has been applied to validate and trial services for mobile networks based on WAP, GPRS and UMTS. Due to the heterogeneity and the complexity of the network elements involved, the classical architectures are not applicable [7]. This is because several entities in the network must cooperate to provide the desired service. For this reason, the authors propose a test architecture based on PCO's and PO's. This architecture is represented in the following figure.

One needs to observe the transit at different strategic points in order to be able to analyse the data exchanged. This leads to the introduction of Points of Observation (PO). A PO needs to be set between the mobile terminal and the gateway. In this way it will be able to collect the data carried by the radio link and the interoperability between the equipment can be verified. The second PO needs to be located between the gateway and the server in order to verify that the data transmitted from the gateway to the server is correct. The Point of Control and Observation (PCO) found upstream to

the mobile device allows initiating transactions and injecting valid, unexpected or invalid events, as well as recuperating the results obtained.

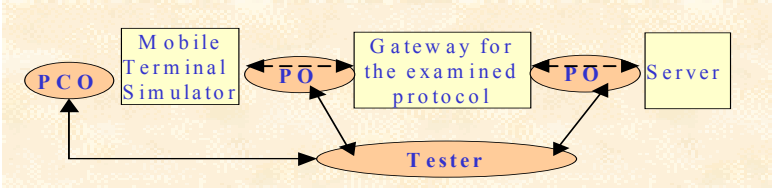

Fig. 2. Generic architecture for testing a mobile application

With this architecture, error detection is powerful and allows capturing all the data exchanged for a given service. If an error occurs between the gateway and the server, the PO will immediately detect it. It is not necessary to place all the PO's but leaving one out reduces the observation capability. There is the possibility of optimising the tests scenarios by removing all the PO's and if a test fails, to reinstall them in order to make a diagnosis and more easily locate the error or errors. This introduces an incremental test methodology based on the degree of test detection capability.

2.3.1 Obtained Verdict

Each PO is coupled with an interface that allows verifying the progress of the test which leads to installing one per zone. In figure 2 we need two interfaces: one between the terminal and the gateway and the other between the gateway and the server. From each PO we will have a file of saved traces. The properties coming from the test objectives will be checked over these traces and in this way errors will be detected and localised. Local verdicts will be pronounced. On the other hand, each PCO will be able to give a local verdict for its zone by using the test scenario of the property being verified. The set of all local verdicts is gathered by a central tester that is in charge of pronouncing the final verdict which will be FAIL, INCONCLUSIVE or PASS.

2.3.2 Test Architectures for the WAP Protocols and Services.

In the beginning, the PLATONIS project partners were interested in the mobile WAP protocol in order to verify that the implemented platform functioned correctly. This protocol was selected because the standard is stable and can be modelled (see next section) and the material that uses it exists and is commercialised.
The test architecture presented in figure 3 is the direct implementation of the model described before.

Above we find the different PO's and PCO's that are responsible for performing the local verdicts. The WAP protocol uses a gateway that transforms the data coming from a WAP navigator (in WML format) to data compatible with the Web servers (in

HTTP format). This gateway thus implements the different protocol layers that are required (WDP/UDP[7], WTP, WSP) in the WAP protocol.

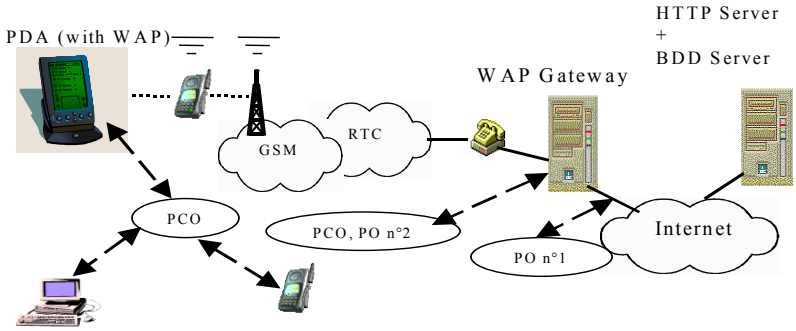

Fig. 3. Test architecture for the WAP

Knowing that the objective is to test a mobile application, it is necessary to obtain the final result on a mobile terminal. For this several options exist:

- Create a secondary network that connects the different testers. In this way one can recuperate the different verdicts without interfering with the application. This solution requires additional equipment: a mobile terminal that can manage simultaneous communications with different mediums. This is not always possible and depends on what protocols are being tested.
- Give the mobile terminal the role of test coordinator and send to it the local results obtained from the different communication mediums that are part of the system under test. This solution does not perturb the test since the results are sent after it is completed, with the bandwidth available to send this data being a client-server type architecture.

  Here we use this second solution for testing the WAP services.

It is also possible to give the gateway the role of remote tester when testing the WAP protocol stack from the client side. In this case one should follow the test architecture of figure 3 and add a PCO at the WAP gateway level. In this way all mobiles can be tested.

3 Functional Verification Model

3.1 Formal Modeling of the Protocol Layers

The modelling of the WAP protocol stack was essentially done for the WTP and WSP layers as well as a simplified version of WAE. This last layer was necessary to be able to dynamically control the system under test. The WAP system was studied for the transmission of messages in connection mode. Following are some important comments on the modelling process:

[7] Wireless Datagram Protocol, often replaced by UDP (Unit Datagram Protocol)

- The WAP standard describes protocols whose behaviour is very complex. This is particularly the case of WTP. This complexity has resulted in 35 000 lines of SDL code for the WTP and WSP layers.
- The abstract ASN.1 notation [2] was used to describe the parameter structures associated to the messages exchanged between protocol layers.
- Effort was made so that the modelling resulted in a modular and flexible one. This was important so that it could be easily adapted to the different test architectures defining where the PCO's and PO's are located. The use of the object oriented SDL 2000 made this modularisation possible.

The specifications were found incomplete and often ambiguous. This was particularly true with respect to error behaviour and the parameters corresponding to the error codes. The specification does not specify and leaves up to the implementation most non-nominal behaviours. For instance, the behaviour of the WAP protocol receiving a signal in a state that did not expect it. This type of behaviour can occur very often due partly to the asynchronous nature of exchanges.

Another case is, for example, when the timer overflows while waiting for a server-side user action. Here, the processes INITIATOR and RESPONDER, that manage the transaction, are killed (return to the NULL state according to the specification). Nothing prevents receiving a signal resulting from an user action, though late, particularly in an asynchronous system. The standard does not give any precisions on this making it possible for different implementations to resolve the problem differently. This could create interoperability problems. Formal modelling of the protocols using languages such as SDL helps eliminating these ambiguities.

3.2 Modeling of the Location Based Services

This section presents the location based services (LBS). These services depend on the geographical positioning of the user terminal and bring a value added to the service offer by the mobile phones. An LBS service can be described in the following way: a mobile phone sends a request to the WAP server via a gateway that transforms the WAP request into HTTP. The WAP server (LCS -LoCation Services - client) sends a request to the LCS server in order to locate the phone. The LCS server lies in the public mobile network PLMN (Public Land Mobile Network). The location of the terminal is calculated using one of the following technologies: GPS, A-GPS, E-OTD, CGI-TA or TOA [13]. Once this location established, the WAP server replies the phone's position.

The knowledge of the user's location allows to develop a new set of applications. The authors have selected and specified, by using SDL, the following services: Nearness Service (for detecting proximity), Itinerary Service (for providing best itinerary to a given place), Emergency Help Service (with location information), Road Traffic Service and Search Service (for searching in a data base associated with the preceding services by using key words).

These services can be integrated on any of the network infrastructures considered (GSM, GPRS and UMTS). However, the methods applied to calculate the user position depends on the type of network. For each network type, different components are

used to perform the calculations. In this section, we first present the SDL description of these services independent of the underlying network. Then, the SDL description of the calculation of the position at the operator level for an UMTS network. The UMTS case is presented here to illustrate this level of description because it has well defined LCS interfaces for both types of network switching modes: circuit and packet modes. For UMTS, the location information is significant and testing its underlying mechanisms is necessary for guaranteeing the reliability of the network.

3.2.1 SDL Description of Services

The services were specified using the SDL-96 description language in order to make it easier to modify the specification by adding or eliminating some functionalities. Data types were defined using ASN.1 allowing the use of variable types such as lists and tables.

The specification of the services was done in two parts. The first part includes the behaviour of the users, the mobile terminals, the application server, the WAP gateway and the LCS server. The second part includes the behaviour of the PLMN for calculating the locations.

Part 1: SDL Description of the LBS Services
The specification consists of four main blocks . Each block describes respectively: the behaviour of the terminal or of several terminals (using dynamic instantiation); the network including the location server and the operator (who gives the temporary terminal id and the SIM id corresponding to the LCS); the WAP gateway; and, the application server (hosting the services introduced before). The SDL specification was developed from an informal description of the services provided by the mobile telephone operators. The modularity followed allows to easily add new services and, in accordance to the services one is subscribed to, modify the subscriber profiles. The specification resulted in a little bit more than 11 000 lines of SDL.

In order to verify that the specification is free from *livelocks* or *deadlocks* [15], the system was simulated using the exhaustive mode.

Part 2: SDL Description of the PLMN Location
In the framework of the UMTS access network and from a LCS [5] perspective, a public mobile network is composed of a core network (CN) and an access network (AN). These interact through the *Le* interface. The GMLC (*Gateway Mobile Location Centre*) is the first node in the CN. The LCS client accesses the network through the *La* interface. The routing information is provided by the HLR (*Home Location Register*) via the *Lh* interface. The GMLC controls the user rights and then transfers the request to the MSC (*Mobile Switching Center*) or the SGSN (Serving GPRS Support Node) via the *Lg* interface. The SRNC (*Serving Radio Network Controller*) in the AN receives the authenticated request from the CN through the *Iu* interface. The RNC (Radio Network Controller) manages the AN's resources (i.e. the LMU's - *Location Mesurement Unit*), the mobile and the calculations. The LMU recuperates the meas-

urements from the signals used for determining the location. These entities communicate by messages sent through the *Iur*, *Iub* and *Uu* interfaces.

The SDL specification of LCS was done taking into consideration the location service architecture as found in UMTS and is briefly described bellow. The system is comprised of two functional blocks [5]. They are:

- The CN block, CoreNetwork, is composed of the processes that describe the behaviour of the GMLC, the HLR and the MSC. In this block, the GMLC process communicates with the HLR and MSC processes through the*Lj* and *Lg* interfaces.
- The AN block, AssessNetwork, is composed of the processes that describe the behaviour of the SRNC and the NodeB.

4 Performance Evaluation

Performance evaluation allows, first, to avoid system malfunctions caused by over-congestion of resources. Second, it allows to identify satisfactory system configurations with respect to some well-defined QoS requirements. System performance engineering has been neglected by software engineers primarily due to the difficulty encountered in using the methods required for performance modelling. It must be noted that 80% of the client/server systems need to be rebuilt due to the lower performance obtained over that required. This should be compared to the cost of performance evaluation that only represents 3% of the total cost of the development. It would be of unique interest to develop tools that better integrate performance engineering in the development process. Particularly in the following aspects: trying to consolidate the link between performance models and functional models; making performance evaluation more accessible to non-specialists, improving efficiency in the development process; and, integrating event simulation techniques. In this last point, it should be noted that design of the simulation model is not expressed using mathematical formulas but by programming. This allows constructing a model as close to reality as possible without making its complete development necessary. It also adapts very well when the objective is to compare different technologies or products or when dimensioning the system being conceived. Other elements, such as the end to end distribution of response time or the loss rate, are very difficult to quantify, making simulation necessary. The performance models for commercial simulators are based essentially on queuing theory. Such a system can be represented by a set of material resources (CPU, buffers, network, ...) accessed by applications (programs or tasks). Eventually, tasks will concur when accessing forcibly limited resources. This type of problem is resolved by scheduling mechanisms that can be associated to the access queues. In the context of the work presented in this article, SES_Workbench [8] was used.

Studying the way to derive a performance model from the functional model was carried out in order to integrate the functional verification with the performance evaluation [6]. MSC (Message Sequence Charts) and SDL (Specification Description Language) are the formalisms that we propose for the specification of the functional aspects. The use of MSC [3] is particularly useful in the case of a service platform where the use of the system can most often be resumed to a limited set of use cases.

Several notations are added to the functional model. These are:

- The *EXEC (uni x, y)* clause on a component means that the computing resources associated with the component are busy for length of time uniformly varying between x and y.
- In a similar way, the *DELAY* clause indicates a delay but concerning external activity to the studied system (i.e. protocol interfaces).
- Other syntax structures allow expressing, for instance, the synchronisation between execution paths, the triggering conditions.

An important point of which we must be aware of here is that a functional model expresses "what a system does" or the functionality it offers, while the performance model describes "the use that is made of resources". Therefore, in the case of the performance model, the functional aspects of the system only appear if they influence the consumption of the resources. The procedure followed here becomes clearer if one considers the following elements:

- When modelling, first one must select the software entities associated to the resources (i.e. the CPU that hosts it) and determine the scenarios that describe the system's common use cases. The software entities are selected according to the desired level of granularity of the results obtained by simulation and of the initial data (unitary measures) that can be procured.
- An exhaustive simulation of a model, such as the location service model, lead to an enormous number of different scenarios. Nevertheless, these scenarios are often redundant from the performance point of view. The choice of one emergency service over another does not modify the performance characteristics unless, of course, one needs to consider different reply times for different service data treatments. Thus, one needs to simplify the model by applying restrictions. This is done by identifying, in the functional model, the external conditions that influence the behaviour of the system being validated, the execution delays and making sure that the data ranges are of pertinence.

The simpler model obtained gives more manageable results when (exhaustively) simulated. Identification of the behaviour needed for performance evaluation can be made as well as the decision branch construction for each behaviour. These decisions are weighted according to the probability and constitute the different performance simulation request types.

In the case of a service platform, the goal of performance evaluation is to improve the configuration sizing with respect to the resources allocated for the different services running on the same platform. For instance, emergency calls should not be affected due to an overload of lower priority services. Figure 4 gives a simplified annotated view of a scenario of an emergency service obtained from the SDL specification described in section 3. Delays are depicted on the instances corresponding to the environment of the platform being validated (here only consisting of a location server). The condition represents a decision branch of the logical behaviour of the service for the scenario.

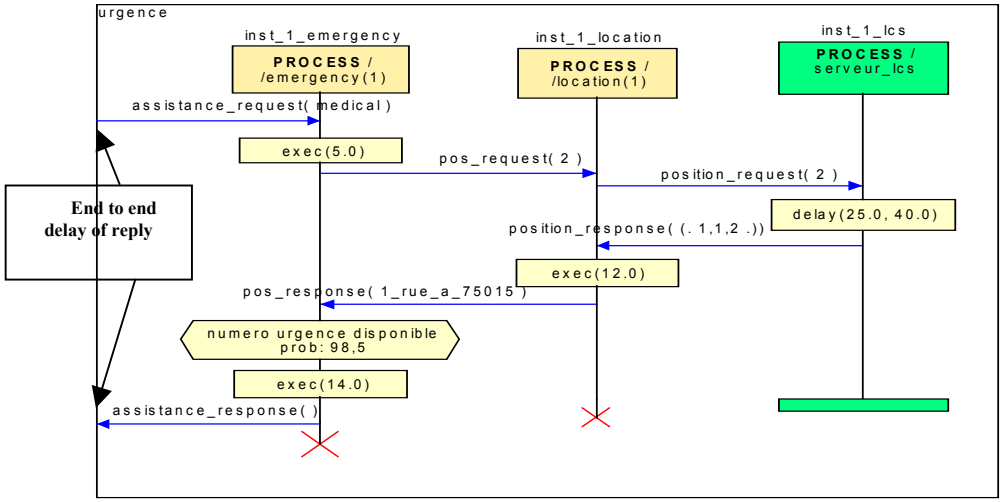

Fig. 4. Emergency Help Service notations

5 Experimental results

5.1 Test generation experiments

Two approaches to test generation of location based applications are presented in this section. The first approach relies on the automatic test generation from the SDL specification of the services and by using a test algorithm. The second approach uses a WML parser to generate the tests from the WML application. The results obtained from the generation of tests for the protocols is also presented in this section.

5.1.1 Test Generation for the *Nearness* Service

In this section we present the test of the *Nearness* service. The scenario generated allows verifying the behaviour of the service in a context with the other services. The test of the other services was performed in a similar manner.

The embedded testing techniques developed at the INT [1] was used for generating the tests of the service without direct access. Starting from the formal specification of the LBS application, a set of test scenarios was automatically generated for the *Nearness* service. To do this, the test objectives were defined. These test objectives are represented in the service specification by a set of transitions to be tested. The method used allows generating test scenarios with the test objectives as a guide for their generation.

This method constructs the test scenario from partial simulations of the specification, to avoid the state's explosion problem.

The scenarios obtained allow to perform conformance tests and to detect errors due to erroneous or unexpected messages. Once the test have been generated and following the architecture defined in section 2, they are applied to the implementation in WLM of the service in order to test its functional behaviour. Only the PCO is taken into consideration during the test of the services, the remaining network is viewed as a black box. The PO's in the architecture are not used for this type of test.

The first step of this method is the definition of the test objectives. For lack of space, here we only give a selection of these objectives that illustrate the behaviour of the Nearness service:

- **Test objective 1**: test to see if the application server makes a request for a position from the LCS server.
- **Test objective 2**: once the location is obtained from the LCS server, test to see if the application server requests from the user a selection from one of the proximity interest points.
- **Test objective 3**: once the user has made his selection, test to see if the application server requests from the user to end the connection or to switch to the *Itinerary* service to obtain the itinerary to the interest point selected.
- **Test objective 4**: test to see if the application server switches from the *Nearness* service to the *Itinerary* service.

Once all the transitions corresponding to the test objectives have been covered, we obtain a test scenario. We have obtained a test scenario that covers all these objectives. It corresponds to the path that has been followed from the environment to the last test objective and has a length of 46 transitions.

5.1.2 Test Generation Method Using an WML Parser

Here we are only interested in the WML application. In this case, the terminal, the gateway and the HTTP server are viewed as a black box. For this, we make the assumption that the information regarding the location is already available and that the network is reliable in the sense that the communication between the HTTP server and the terminal works correctly. The test architecture followed is the one presented in section 2, using only the PCO on the mobile terminal side. The procedure to test the WML application is as follows:

- Generation of the automata for the WML application: The first step is to generate the automata representing the behaviour of the application. For this, a tool called *GenTree* [5] was developed. This tool takes as input the WML application, performs a lexical and syntactical analysis, generates a behaviour tree, visualises it and saves it in the form of an automata.
- Conversion of the automata to SDL: The second step is to take the behaviour automata and convert it to SDL. Work on how to transform an EFSM (Extended Finite State Machine) to SDL is described in [11]. Using tools available on the platform, the resulting SDL description is used to automatically generate the tests.

5.1.3 Test Generation for the WSP and WTP Protocols

As explained before, the WTP and WSP layers are not directly accessible. Therefore, to test them we must also use the embedded testing techniques. The tests also allow testing the interoperability between the protocol layers.

For the generation of the tests, test objectives were defined based on the WTP and WSP specifications from the WAP Forum [10]. The following table gives some insight on the results obtained for the test generation of the standards.

Table 1. Test objectives for the WSP and WTP layers

	Test Objectives	**Test scenario length (in n° of transitions)**
WSP	Session connection and disconnection phase	53
	Refused session connection phase	121
	Moved session connection phase	29
	Complete transaction phase	98
	Transaction abortion phase	144
	Session suspension and reactivation phase	93
	Session suspension and refused reactivation phase	93
	Session suspension and termination phase	67
	Complete PUSH transaction without confirmation phase	60
	Complete PUSH transaction with confirmation phase	63
	PUSH confirmation interruption phase	68
WTP	Basic transaction class 0	52
	Basic transaction class 1	100
	Basic transaction class 2 without validation from initiator	135
	Basic transaction class 2 with validation from initiator	38
	Basic transaction class 2 with interruption from initiator	67
	Basic transaction class 2 with interruption from replier	69

5.2 Test architecture experiments

In this section we present the deployment of the different PO's and PCO's that make up the test architecture (figure 3, section 2). Each PO is made up of two parts: PO_trace for traffic inspection and PO_analysis for giving the local verdict.

In general, the WAP gateways are connected to Internet via a local network. In this type of network, all the machines can capture the exchanged data. Thus, the PO_trace can be based on a "sniffer" that will not perturb the network. For portability reasons, the "sniffer" was implemented in Java using *Jpcap*. *Jpcap* is based on *Winpcap* for Windows and *Lipcap* for Unix. On the other hand, PO_analysis has a set of defined properties that it needs to verify. It will verify them on the trace provided by PO_trace. Once the verification is done, PO_analysis will produce the local verdict and send it to the tester that centralises all the verdicts.

The PO n°2 (see Figure 3) observes the traffic received and emitted by the WAP gateway. The open source Kannel gateway was used allowing the code modifications

needed for installing the trace tools. The Kannel software is structured as different layers, each implemented as a thread that communicates with the others by exchanging messages. Following this architecture, PCO's have been located between the different layers. The installed PO is made up of a PO_trace_in that recuperates incoming traffic, a PO_trace_out that recuperates outgoing traffic and a PO_analysis that inspects all the traces and gives out the local verdict. These PO's can be used to test the behaviour of the gateway when there is no possibility of incorporating the PCO inside the gateway. The information recuperated by the PO's is very descriptive and includes the name of primitives, the data transferred, the states reached, etc. Figure 5 depicts a sample of the trace obtained from the WSP layer PO. The WSP layer is the one that allows setting up and releasing a session between the client and the server using one of the two connection modes. In figure 5 one can see the connection between the client (the PCO program running on the PDA: *PDA_Tool-Kit*) and the server (with the IP address *157.159.100.113*) using the connection oriented mode (*9201*). The message *TR-Invoke.ind* is used to open the *welcome.wml* page.

```
2003-04-22 11:23:43 [1] INFO: From WTP: Primitive Name: TR-Invoke.ind
2003-04-22 11:23:43 [1] INFO: From WTP: Ack Type: 0x01
2003-04-22 11:23:43 [1] INFO: From WTP: WTP Class: 2
2003-04-22 11:23:43 [1] INFO: From WTP: WAPAddrTuple 0x823e5d0 =
<157.159.100.113:2761> - <0.0.0.0:9201>
2003-04-22 11:23:43 [1] INFO: From WTP: Handle: 9
2003-04-22 11:23:43 [1] INFO: WSP Get PDU at 0x821d908:
2003-04-22 11:23:43 [1] INFO:  GET, OPTIONS, HEAD, DELETE, or TRACE: 0
2003-04-22 11:23:43 [1] INFO:  Length of URI: 28
2003-04-22 11:23:43 [1] INFO:  URI:
2003-04-22 11:23:43 [1] INFO:  Octet string at 0x8217f70:
2003-04-22 11:23:43 [1] INFO:    len:  28
2003-04-22 11:23:43 [1] INFO:    size: 29
2003-04-22 11:23:43 [1] INFO:    immutable: 0
2003-04-22 11:23:43 [1] INFO:    data: 68 74 74 70 3a 2f 2f 6c    http://l
2003-04-22 11:23:43 [1] INFO:    data: 6f 74 69 3a 38 30 30 30    oti:8000
2003-04-22 11:23:43 [1] INFO:    data: 2f 77 65 6c 63 6f 6d 65    /welcome
2003-04-22 11:23:43 [1] INFO:    data: 2e 77 6d 6c                .wml
2003-04-22 11:23:43 [1] INFO:  Octet string dump ends.
2003-04-22 11:23:43 [1] INFO:  Request headers:
2003-04-22 11:23:43 [1] INFO:  Octet string at 0x82448a0:
2003-04-22 11:23:43 [1] INFO:    len:  230
2003-04-22 11:23:43 [1] INFO:    size: 231
2003-04-22 11:23:43 [1] INFO:    immutable: 0
2003-04-22 11:23:43 [1] INFO:    data: 81 83 81 84 81 85 81 86    ........
.....
2003-04-22 11:23:43 [1] INFO:    data: 74 2d 72 65 73 70 6f 6e    Projet_P
2003-04-22 11:23:43 [1] INFO:    data: 73 65 00 80 9e a9 4e 6f    latonis/
2003-04-22 11:23:43 [1] INFO:    data: 6b 69 61 2d 4d 49 54 2d    PDA-Tool
2003-04-22 11:23:43 [1] INFO:    data: 42 72 6f 77 73 65 72 2f    -Kit/V1.
2003-04-22 11:23:43 [1] INFO:    data: 33 2e 30 00 83 99          0.......
2003-04-22 11:23:43 [1] INFO:  Octet string dump ends.
2003-04-22 11:23:43 [1] INFO: WSP PDU dump ends.
2003-04-22 11:23:43 [1] INFO:

2003-04-22 11:23:43 [1] INFO: From WTP: Primitive Name: TR-Result.cnf
```

Fig. 5. WSP PO in the Kannel WAP gateway

The PCO must be capable of sending and receiving different frames as well as the different local verdicts. A PDA running Windows CE was used making it easier to program and establishing either a direct connection to GSM or through a mobile phone equipped with a *Irda* port. A WAP navigator was developed that implements the WTP and WSP layers (standard WAP version 1.1) and provides a graphical user interface that allows loading the test files.

While the test is being performed, the PDA waits for data that is either information feedback or a local verdict. At the end of the test, it waits until it receives all the verdicts and only then produces the final one based on the rule stated previously. Figure 6

bellow shows the beginning of a test, including the request for the connection of figure 5 but seen from another perspective.

Also, to test the protocols found on the client side (mobile), a PCO has been installed at the WAP Kannel gateway level. The test scenarios produced as presented in section 4 are executed on the platform using the PCO at the gateway level, but also using the PDA as previously described.

To give an example, a test scenario produced as presented in section 5.3 will be described. This scenario is injected at the PCO level and allows testing the "class 2 transaction with interruption from replier". As seen in figure 7, one can observe the PO level exchanges. In this way we are able to check that the mobile behaves correctly when an unexpected message is received during the exchanges. Resulting from this injected error, the existing WML page (that the terminal wanted to access) cannot be opened due to the message sent via the PCO.

Fig. 6. WSP PCO at the PDA

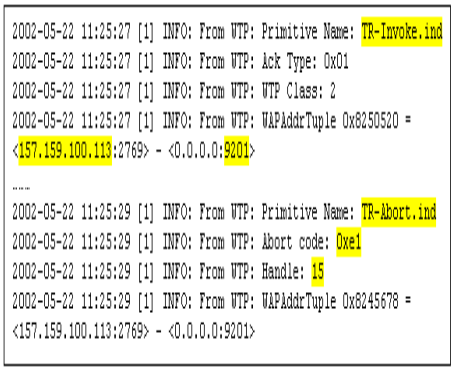

Fig. 7. WSP PO with injected message

6 Conclusion

The platform presented in this article allows performing conformance and interoperability tests, analyse the specification and produce a performance model for the services. It covers all of the validation phases: formal specification, test architecture definition, test generation and execution for the given architecture and performance evaluation. Its originality lies in its ability to cover several different aspects that start with the formal specification and end with the incremental implementation of the tests following an innovative test architecture. It has allowed to experiment with embedded test techniques and is designed to allow end-users to test real applications in their environment. It also shows the scalability of the proposed methods.

Complementary research on several aspects is being carried out. The test of an WML application is currently based on a behaviour tree extracted from the specification. To be able to use automatic generation tools, this tree is translated to SDL. Calculating test coverage also becomes necessary. Introduction of control and observation

events in the specification is being studied in order to increase the testing capability of an application.

The complexity of the architecture must be considered in the performance modelling. The architecture is composed of many components and is always evolving (i.e. adding/removing services, evolution of standards). Thus, another important aspect that is being studied is the component modelling and composition rules and how they take into account performance and resources.

Finally, the techniques presented here will be applied to other types of mobile networks, as for instance, ad-hoc networks. This type of networks do not require (or preferably don't depend on) a centralised management or a fixed network infrastructure, such as, base stations and fixed access points. Furthermore, dynamic reconfiguration of the network becomes necessary in order to adapt to eventual context changes. These new characteristics make it necessary to adapt existing test methods and devise new ones.

References

1. A. Cavalli, D. Lee, Ch. Rinderknecht, and F. Zaïdi. Hit-or-Jump: An Algorithm for Embedded Testing with Applications to IN Services. In Proceedings of FORTE/PSTV'99, Beijing, China, Octobre 1999.
2. O.Dubuisson. ASN.1. Springer, 1999.
3. ITU-T, Message Sequence Chart (MSC), Recommendation Z.120, November, 1999, http://www.sdl-forum.org
4. ITU-T, Specification and Description Language, Recommandation Z.100, Nov. 1999, http://www.sdl-forum.org
5. M. Mackaya, R. Castanet. Modelling and Testing Location Based Application in UMTS Networks. IEEE Contel, Zagreb, Croatia, June 2003.
6. W. Monin, F. Dubois, D. Vincent , P. Combes, Looking for a better integration of design and performance engineering, SDL Forum 2003.
7. O. Rafiq, R. Castanet and C. Chraibi. Towards an environment for testing OSI protocols. Proc of the International Workshop on Protocol Specification, testing and Verification, Toulouse, France, 1985.
8. SES Inc. SES WorkBench Modelling Reference manual, 1998.
9. The PLATONIS Consortium. The platonis project. In First International Workshop on Services Applications in the Wireless Public Infrastructure, Mai 2001. http: //www-lor.int-evry.fr/platonis.
10. WAP spécification, http://www.wapforum.org.
11. YoungJoon Byun, Beverly A. Sanders, Chang-Sup Keum, Design Patterns of Communicating Extended Finite State Machines in SDL, PloP 2001 conference
12. http: //www.kannel.org.
13. http://www.wirelessdevnet.com/channels/lbs/features/mobilepositioning.html.
14. http: //www-lor.int-evry.fr/platonis
15. http://www.telelogic.com