COMPLIANCE OF WIRELESS APPLICATION PROTOCOLS

Ousmane Koné
LaBRI/CNRS - INPL***
* *351, Cours de la Libération F-33405 Bordeaux*
** *2, Avenue de la Forêt de Haye F-54516 Nancy*
kone@labri.fr - okone@ensem.inpl-nancy.fr

Abstract The work presented in this paper contributes to WAP testing effort, by proposing an approach to compliance tests development. Here, this approach is illustrated with the design of compliance or interoperability tests for the WSP-protocol operating over a WTP-service.

Keywords: Compliance, Interoperability, Wireless Systems, WAP, Internet, WWW.

1. INTRODUCTION

Wireless Application Protocols (WAP) constitute reference standards for the new architectures combining mobile telephony and Internet. In order to achieve reliability and interoperability, WAP implementations must be tested against these standards. According to the WAP forum, the testing process would follow two tracks/phases: track one, *Certification testing*, focused on WAP application level, and track two, *Compliance testing*, concerned with protocol level. To date, the main developments have been concerned with track one and since last year, more than 60 products from industry (mobile equipments and servers) have been WAP-certified. But we are not aware of developments on compliance issues which are the concern and contribution of our paper.

Some results of our work will be used in connection with a French RNRT project [11], in which we have been asked to propose test platforms for experimenting new industrial applications in wireless environment. Here, a testing approach to compliance of WAP sublayers, is proposed and illustrated with WSP (Wireless Session Protocol) layer.

Section 2 recalls some main issues related to WAP systems compliance. Section 3 provides a brief description of our testing approach. In

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35497-2_31
I. Schieferdecker et al. (eds.), *Testing of Communicating Systems XIV*

section 4, we comment the results of this approach to the design of tests against WSP operating over a Wireless Transaction Service.

2. WAP SYSTEMS AGAINST COMPLIANCE

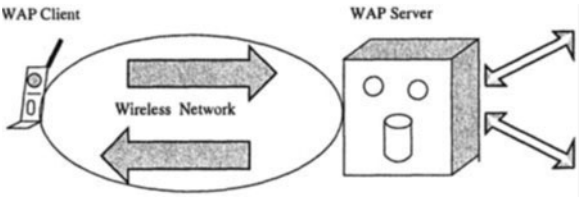

Figure 1. Wireless Network

Within the WWW, a computer program can ask another one to execute some request. The first program is called *client* and the second one is called *server*. In a similar manner, WAP is aimed at enabling wireless terminals to access Internet-like services (figure 1). WAP features are organized into different layers (figure 2). The **Application layer** provides the wireless terminal with a uniform environment enabling access to standard telephony and WWW services. This application environment includes a WML browser. The **Session service** provides application layers with a consistent interface for maintained session. We will be back to this layer in section 4. The **Transaction layer** handles the execution of transactions requested by upper layers, in a transparent manner. etc.

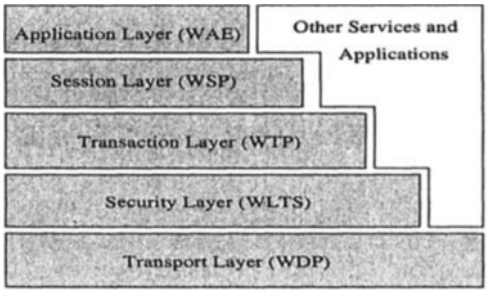

Figure 2. WAP protocol stack

For interoperability reasons, WAP systems must respect the above standard. To date, WAP forum has concentrated its efforts on the development of a certification program, which consists of validating WAP

products application environment. The ICS (Implementation Conformance Statements) constitute reference documents on the basis of which manufacturer products are certified for conformance. Certification testing is carried under the responsability of an organization sanctioned by WAP forum as the "Certification Authority". The components involved in a test campaign are WAP-certified equipments, and the framework and process for delivering certification labels are defined by WAP forum. From November 2000 to November 2001, certification labels have been delivered for more than 6 WAP servers and more than 55 WAP terminals (clients). But there are few developments published on the testing of underlying protocol operations. Compliance involves protocol *inter-operations*, in the sequel, we will interprete *compliance testing* as some *interoperability testing*, and vice versa.

3. TESTING APPROACH OVERVIEW

The test development approach adopted follows a generic methodology that we named TOP [5, 1, 6], and which handles interoperability test automation from specifications to test execution via a CORBA platform. For space reasons, we will not racall all aspects of TOP in this paper. We invite the reader to report to the above references for details. In the following, we consider some key issues of standard testing activity:
(•) *Test architecture:* The test architecture represents the configuration in which a test experiment is undertaken. (•) *Test case generation:* A test case (≡ test sequence) is a sequence of interactions to be submitted to implementation under test. (•) *Test deployment:* Test deployment is concerned with effective execution of test cases within a given test architecture. Even if the literature is poor in this field, test deployment remains an important aspect of testing activity.

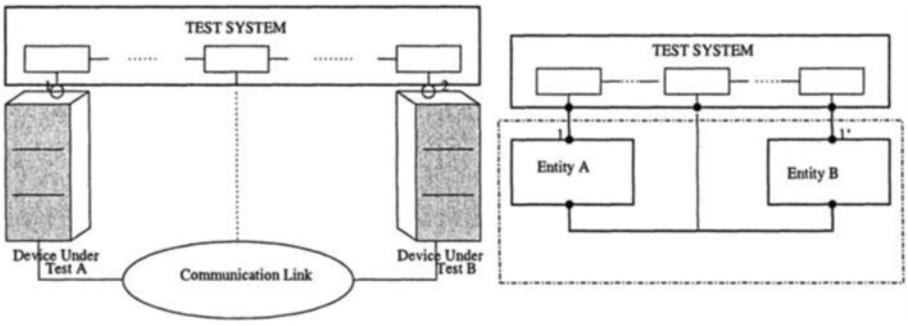

Figure 3. Functional views of the Test Architecture

Test Architecture and Deployment. Figure 3 depicts functional views of the architecture we use for interoperability. The test system interacts with the DUTs (Devices Under Test), through the Service Access Points (also corresponding to PCOs: Points of Control and Observation) labeled with 1 and 2 (1 and 1' in the right part). An optional interaction point enables it to access the PDU (Protocol Data Unit) at lower communication level. Notice that this architecture is similar to the one proposed by the ATM forum for interoperability. The monitor point of ATM forum corresponds to the optional interaction point in figure 3. In our approach, implementations can be stimulated, directly with the test cases executed by the test system through PCOs, or via upper layer running-applications (figure 4). In both cases, the test sequences computed constitute reference traces that can be checked during or after test execution (kind of passive testing). But those architectures, including the ones defined in standard ISO-9646 [3], describe only functional configurations. In this case, they do not provide information to concrete design of PCOs for the interaction between the test system and DUTs. This aspect is left to test laboratory and test execution concern. In our methodology, we have designed a portable test platform for handling test deployment issues [6]. The platform is featured over the CORBA standard, which facilitates test deployment for a wide range of DUTs. We do not develop test deployment aspects in this paper.

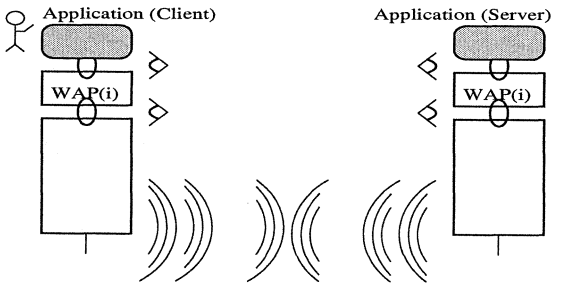

Figure 4. Towards an experimental test configuration

Test case generation. Unlike usual techniques to interoperability test sequences generation [2, 10], our test generation tool implements an *on the fly* exploration technique, which is recognised to be efficient against complex specifications [4]. It directly extracts test paths by analysing the sub-specifications (model of entity A and model of en-

tity B). The algorithm implemented works with input/output automata modelling the specifications and the TPs (*Test Purpose* $\stackrel{\text{def}}{=}$ *A behavioral property to be tested*). It has been implemented in the so-called TESCOMS tool by our students, with C++ language. The tool has been successfully used before, with specifications like ATM sublayers, and we are currently using it for computing WAP specifications.

This section summarized the main steps of our testing approach: (`Specifications`) $\rightarrow$ TESCOMS $\rightarrow$ (`Test case`) $\rightarrow$ TBROKER $\rightarrow$ ... ... $\rightarrow$ TBROKER $\rightarrow$ (`Test Deployment/Execution`). Below, we consider the case study with WSP.

4. COMPLIANCE TESTS SUITE PRODUCTION

Following the general recommendations of WAP forum, on certification and compliance testing process [8], we have shown in previous works, the feasability of automatic tests computation for WAP. Here we focus on reporting our test production experiment against the session layer.

Specification and Test Purpose issues. WSP [9] enables client and server applications to exchange contents through a maintained communication session. Here, we consider the connection-mode protocol only, as the connectionless one is fairly simple. WSP is mainly featured to establish and release a session between a client and a server, exchange contents of the two applications, suspend and resume the session. In the specification, service primitives are of the form `S-Service-type`, (e.g. `S-Connect-req`), where `S` is the label of the session layer, `Service` indicates the name of the service. The type is also indicated. Protocol inter-operations are performed through the use of a transaction service (`TR`). For remembering the WAP stack, look again at figure 2. The PDUs exchanged between the WSP peer-entities are encapsulated in the `TR` primitives: `TR-Service(PDU)` denotes the encapsulation of a given `PDU` in a `TR-Service` primitive. In the standard, the session protocol is described with a set of state tables that we exploited and translated directly in terms of input/output automata and SDL specifications. Of course, we will not include the specification model (equivalent of about 3000 lines of SDL code) in this paper.

The reference document we used for selection of test purpose, was the WAP ICS [8]. WSP implementation features are organized into groups of functionality: `Session creation`, `Session suspend/resume`, `Push facilities`, etc. In this paper, we will not include all the test cases

considered. We will present a couple of examples to illustrate the experiment results: the session creation - the session release.

The test suite / Session Creation - Session Release. These features fall in the category of mandatory services to be implemented by both WAP clients and servers.
Test method: The session creation starts with an `S-Connect-req`, and ends with an `S-Connect-cnf`. The session release starts with an `S-Disconnect-req` (initiated, in this example, by the server) and ends with the notification of the disconnection.

The test purpose is defined with a finite automaton: The syntax used requires one transition per line. Each line describes an interaction as well as the implementation which executes it. In the first line of the example below, "1" represents the identifier of the WAP-client implementation. In the second line, "2" represents the identifier of the WAP-server implementation. The last state encountered (`TP7` in the example) is implicitely an accepting state.
Test Purpose: Identifiers `WSP_CO_C001`, `WSP_CO_C002` of the ICS proforma.

```
TP1 ? 1.S-Connect-req 1 TP2
TP2 ? 2.S-Connect-res 2 TP3
TP3 ! 1.S-Connect-cnf 1 TP4
TP4 ? 2.S-Disconnect-req 2 TP5
TP5 ! 1.S-Disconnect-ind(DISCONNECT) 1 TP6
TP6 ! 2.S-Disconnect-ind(USERREQ) 2 TP7
```

The output below represents the test case produced by the TESCOMS test generation tool. The comments of the form `[XXX XXX XXX]` represent internal identifiers related to the global state which may be actually reached during the computation of the concurrent input system: In the first line example, `NULL` represents the starting state of the client side, the second `NULL` represents the starting state of the server side, and `TP1` represents the beginning of the test purpose. Between these identifiers are displayed (with tabs) the actual interaction to be executed.
Test case dynamic behavior:

```
[NULL NULL TP1]
 1:    ? 1.S-Connect-req 1
[NUCIA1 NULL TP2]
 2:     ! TR-Invoke(Connect) 1
[CONNECTING NUCIA1 TP2]
 3:      ! TR-Invoke.ack 2
[CONNECTING NUCIA2 TP2]
 4:       ! 2.S-Connect-ind 2
[CONNECTING CONNECTING TP2]
 5:        ? 2.S-Connect-res 2
[CONNECTING CIC2A1 TP3]
 6:         ! TR-Result(ConnectReply) 2
```

```
[CICEA1 CONNECTING-2 TP3]
 7:          ! TR-Result.ack 1
[CICEA2 CONNECTED TP3]
 8:           ! 1.S-Connect-cnf 1
[CONNECTED CONNECTED TP4]
 9:            ? 2.S-Disconnect-req 2
[CONNECTED CENUA1 TP5]
10:             ! TR-Invoke(Disconnect) 2
[CENUC1 CENUA2 TP5]
11:              ! 1.S-Disconnect-ind(DISCONNECT) 1
[NULL CENUA2 TP6]
12:               ! 2.S-Disconnect-ind(USERREQ) 2
[NULL NULL TP7]
```

Test case analysis: [1:] A Connection request is received by implementation 1 (the client session entity). [2:] The client invokes the transaction layer for sending its `Connect` PDU. [3:] Upon reception of that PDU, the server (session entity) acknowledges it through the transaction layer. [4:] The server sends a Connection indication to the upper layers. [5:] Then it receives the response notifying that the application accepted the connection. [6:] This notification is sent to the client, as a result of its `Connect` request. [7:] The client acknowledges the result. [8:] The client session user is sent a confirmation notifying that the connection has been established. [9:] The server side receives an Disconnection request. [10:] Then the request is hold by the transaction layer. [11,12:] The upper layers are notified that the session has been released.

Analysis of the produced test suite. Table 1 recaps the features covered by the overall WSP computed test suite. In this table, the column "Function" represents service functionalities. For each function, we present the PDUs involved in the operation of the function, or the related capabilities. Some of the functionalities specified in the standard are mandatory while the other ones are optional. Most of the mandatory features have been computed (80%). Capabilities represent a set of service facilities and parameter settings related to the operation of the service provider. There are different kinds of facilities defined in the standard, but the service provider may recognise optional/additional facilities. Some examples of facilities are *Aliases, Extended methods, Protocol options.* The basic methods of WAP correspond to the ones defined in the HTTP/1.1 standard, and the *Extended methods* correspond to those beyond HTTP/1.1. *Proprietary methods* fall in this category. During the connection, the peer entities can perform a *Capabilities negociation* in order to agree on a common communication profile. The `Connect` PDU can be used to specify the required capabilities of the initiator, and the `ConnectReply` PDU would contain the capabilities acknowledged by

Table 1. Analysis of dynamic behaviour computed

Function	PDU/Capabilities	Mandatory Feature	Tests computed
Session Creation	Connect PDU	Yes	Yes
Capabilities Negotiation	Connect PDU, Connect Reply PDU	Yes	No
Session Release	Disconnect PDU	Yes	Yes
Session Suspend, Session Resume	Suspend PDU, Resume PDU	No	Yes
Push	Push PDU	No	Yes
Confirmed Push	ConfirmedPush PDU	Yes	Yes
Extended Methods	Proprietary Methods	No	No
Methods GET, POST	Get PDU, Post PDU, Reply PDU	No	Yes
Methods DELETE, HEAD, OPTION, TRACE	Get PDU, Reply PDU	No	Yes
Method PUT	Post PDU, Reply PDU	No	Yes

the responder. Since all the actual capabilities are not provided in the current standard, and as these capabilities depend on the actual WSP service users, we did not look into this aspect. For the time being, we have treated the connection procedure (session creation PDUs) that conveys users defined facilities. 100% of the standard defined PDUs and also 100% of the standard defined ASPs are involved in the test suite. This guarantees that each elementary protocol operation can be experimented by the computed test suite.

In the overall test suite, some portions of tests suite are replayed. For instance, all the `Method Invocation` facilities have basically the same operation scheme. At the WSP level, methods can be invoked by three basics PDUs, namely the `Get PDU`, `Post PDU` and the `Reply PDU`. The current `Method` reference is encapsulated in these PDUs.

5. CONCLUSIONS

The work presented is a contribution to the testing of protocol interoperations against compliance with WAP standard. Of course, for space reasons, we could not include in this paper all the test cases designed. We illustrated our work with WSP, which is the first layer below application level. Almost all the mandatory functionalities have been experimented in dynamic test generation. We did not complete the aspects related to capability negociation, which are rather a concern of actual telecommunication service provider. Furthermore, the number of possible facilities is unknown a priori. A preliminary static interoperability review of actual WAP-applications communicating through the WSP service may allow for the selection of a subset of relevant facilities to be experimented.

Our testing approach, suitable for layered protocols, can be experimented with other lower layers of WAP stack, with few limitations. For the time being, we are interested in concrete experimentation/deployment of the produced test cases. Most of telecommunications operators have already done some developments related to the WAP technology, and we have obtained, for free, a prototype of WAP client, and a WAP server (with full source code in C language). We are currently using these implementations to conduct concrete test experimentations.

References

[1] X.Balliet, N.Eloy. Development of a distributed test system with CORBA. Co-authored Master's thesis. ENSEM, Polytechnic Institute Nancy, March 2000.

[2] A.Fukada et al. A conformance testing for communication protocols modeled as a set of DFSMs with common inputs. In. Testing of Communicating Systems Vol. 10. Chapman & Hall, 1997.

[3] ISO/IEC 9646, Information Technology – Open Systems Interconnection-Conformance testing methodology and framework - Part 1-5. 1991.

[4] O.Koné. A local approach to the testing of real-time systems. *The Computer Journal*, British Computer Society, Oxford Press. Vol. 44 N.5, 2001.

[5] O. Koné, R. Castanet. Test generation for interworking systems In. *Computer Communications*, Elsevier Science Publishers, Vol. 23 N.7, 2000. pp 642-652.

[6] O.Koné. The TBROKER platform for the interoperability of communications systems. 5^{th} IEEE/Worldses Conference on Communications. Crete, July 2001.

[7] Wireless Application Protocol. Architecture Specification, WAP forum, April, 1998. URL: `http://www.wapforum.com`

[8] Wireless Application Protocol. Conformance Statement, Compliance Profile and Release List. WAP forum, April 1998. URL: `http://www.wapforum.com`.

[9] Wireless Session Protocol Specification. WAP forum.
URL: `http://www.wapforum.com`, 1999

[10] M.Malek, S.Dibuz. A pragmatic method for interoperability test suite derivation. Proc. EUROMICRO Conference. Sweden, August 1998.

[11] Réseau National de Recherche en Télécommunications. URL: `http://www.telecom.gouv.fr/rnrt`.