Acta Cybernetica 15 (2002) 653–667.

Standardized Event Pair Based Test Generation Method Using TSS&TP

Zoltán Pap,* Zoltán Réth//áti,* Róbert Horváth,* and Gusztáv Adamis*

Abstract

In the software engineering test development takes significant resources. A general method for the creation of appropriate test suites could solve the problems of the often ad-hoc and time-consuming test generation process. The recent method uses formal specifications to support systematic derivation of complete test suites. From the formal specification using a special procedure a formalized document, the so-called Test Suite Structure (TSS) and Test Pur-poses (TP) can be created. With the help of this document developers can easily, automatically implement the test suites. The TSS&TP document also enables the persons who perform the tests to under-stand the test criteria and the steps, even if they do not actually know the protocol itself. We present a thorough picture of our test derivation method and show its efficiency on the Wireless Transaction Protocol (WTP) of the Wireless Application Protocol family (WAP). During our work in the validation phase we also found some operational flaws in the protocol specification.

Keywords: Test generation methods, Formal description, Test Purpose, Test Suite Structure, Validation, WAP, WTP

1 Introduction

Conformance testing is the process for checking whether the dynamic behavior of the already implemented protocol is in conformity with the standard. There are idealized requirements of conformance testing [1]. The test should cover the whole protocol (check all the possible functional behaviors for different event sequences - this is measured by the so-called coverage value which means what percentage of the graph of the extended state space of the automaton the test verifies). The test should also check abnormal situations, observe reactions to improper events. The test suite should be so created that the elements of it, i.e. the test cases, could be executed separately. Meeting all these requirements is a big challenge for all participants of the telecommunication field. The standardization institutes, the

*Budapest University of Technology and Economics Department of Telecommunications and Telematics H-1521 Budapest, Hungary,
e-mail: {pap, rethati, horvath, adamis}@ttt-atm.ttt.bme.hu

equipment manufacturers and service providers all could benefit from test generation methods that are standardized and easy to derive.

Usually highly qualified experienced engineers carry out the test generation process. They write the tests directly in C or in formal test language, for example in Tree and Tabular Combined Notation (TTCN), without any inner step directly from the standard, which is mostly an informal textual description. This knowledge-based, time consuming work leads to high costs, has lower reliability and is not always complete. On the top of all that it begins with a long learning phase when the developers have to get deeply acquainted with the new protocol. We call this way of test creation the "traditional method" in this paper.

The other known way of test generation is the computer aided automatic method [14][5]. This has also longer historical background, but until now there have been no real public solutions or efficient algorithms to provide satisfactory coverage by reasonable amount of test cases. The generated unstructured test suite is hard to execute. These automatic methods require a full formal specification as input.

We worked out a special method for the test generation process of conformance test cases that tries to mix the advantages of both previously mentioned methods.

The formal specification provides valuable information for the protocol, and completely describes the behavior of the automata. In the course of our work we implemented the WAP WTP [9][8] in a widely used formal description language, in the Specification and Description Language (SDL). We created a test suite to it starting from the SDL description and using our new method. We rationally built up a test suite structure, in which the enhanced test purposes of systematically divided and chosen elementary test cases are put. The enhanced test purposes contain not only the textual description of the purpose of the test case, but also the main information and properties of the test case in a regular form. The result of this procedure is a regular and formal document that we call Test Suite Structure (TSS) and Test Purposes (TP). This TSS&TP document contains the necessary information of the whole test suite. As each test purpose represents the description of one test case, the TTCN, C, or other code representations can be implemented easily, automatically. We present the whole method on the WAP WTP, and compare its properties to the traditional and computer aided solutions.

In the frame of this paper in the last chapter we note that in the model of WAP WTP we found flaws, which could - under special circumstances - possibly cause problems in the operation of the protocol [11].

2 Specification and Description Language

The Specification and Description Language (SDL)[12] is a formal language, which is widely used for specifying especially telecommunications systems. The formal description technique SDL is standardized by ITUT as Recommendation Z.100. In general one can choose different approaches describing systems. SDL puts emphasis on the behavior of these entities including data flow. Other fields of describing systems are out of the scope of the language. The development of SDL started in

1972 after observing the requirements of describing different complex systems. The first version was released in 1976, and new versions followed in each four years. One of the strengths of SDL is that it is a well-accepted world standard. It is supported by the ITU-T (CCITT) and ISO, thus it can be used independently of the different companies.

The typical properties of systems that can be effectively described in SDL are:

- The types of the systems observed can be real time or interactive
- The domain of the observation can be observation of behavior or observation of architecture.
- Level of abstraction can range from overview to details.

The preciseness of the SDL description makes possible to compile the description to other lower level languages such as JAVA or C. This process can be fully automated, so it is possible to decrease the time needed for developing systems and it is also possible to guarantee the correct behavior. And the SDL diagrams fully comply with the documentation and make it possible to maintain and develop the system easily.

2.1 The SDL system

The main object in the SDL abstraction is called system. This is the formal model of an existing or planned real system. Everything not belonging to the system is called environment. System can be open or closed (it depends on whether it has connection with its environment). Such a system is a collection of SDL processes which communicate asynchronously by exchanging messages. The reception of a message may force a process to change its state. During such a state transition, the SDL process may send new messages and/or perform operations on local variables. SDL processes are combined to (sub)systems by means of block diagrams. In a block diagram, the process specifications are referenced and the communication links among the processes and between the processes and the system environment are defined. All of the processes have their own memory for storing their own variables and state information, and all of them contain a FIFO buffer of infinite length, called queue, for the incoming signals. The process reads the signals from this queue on order of coming (this does not apply to priority signals). The process takes the signal in the first position in the queue, if the process has a predefined behavior for the signal, then it reacts accordingly, otherwise the process ignores the signal and moves on to the next one.

3 The WAP WTP

3.1 The WAP

The Wireless Application Protocol is set of protocols that operate over wireless communication networks. These protocols are designed for wireless devices such as mobile telephones, pagers, and personal digital assistants. The specifications extend mobile networking technologies (such as digital data networking standards) and Internet technologies (such as XML, URLs, scripting, and various content formats).

The WAP consists of layers that more or less correspond to the OSI layers. Each of the layers of the architecture provides services for the layers above. The WAP architecture can be seen in Figure1.

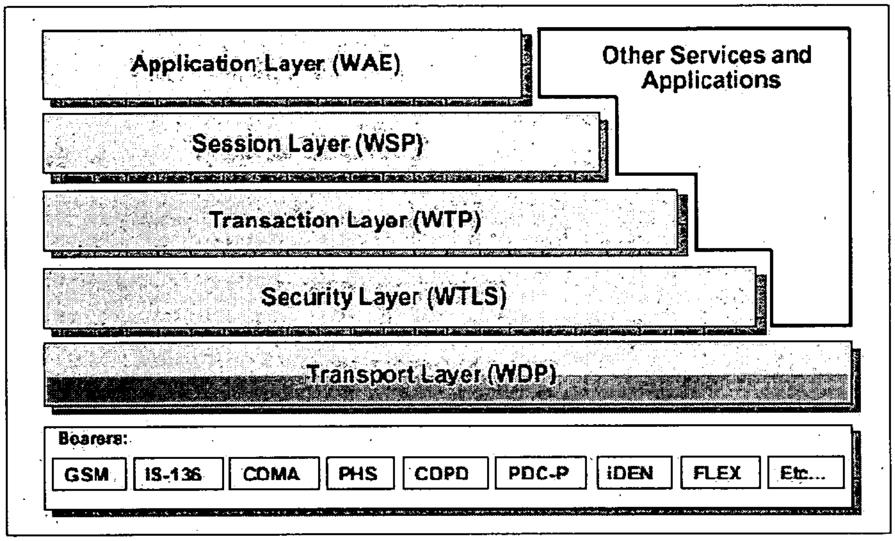


Figure 1: The WAP Architecture

3.2 The WTP

WTP is the transaction protocol of WAP. During a browsing session, the client requests information from a server, which may be fixed or mobile, and the server responds with the information. The objective of the protocol is to reliably deliver the transaction while balancing the amount of reliability required for the application with the cost of delivering the reliability. WTP runs above a datagram service and optionally a security service. The main features of the protocols are:

- Improved reliability over datagram services. WTP relieves the upper layer from re-transmissions and acknowledgements, which are necessary if datagram services are used.
- Improved efficiency over connection oriented services. WTP has no explicit connection set up.
- WTP is message oriented and designed for services oriented towards transactions, such as "browsing".

The WTP operates efficiently over secure or non-secure wireless datagram networks. There are three classes of transaction service:

- Class 0: unreliable invoke message with no result message (unreliable one-way requests),
- Class 1: reliable invoke message with no result message (reliable one-way requests),
- Class 2: reliable invoke message with exactly one reliable result message (reliable two-way request-reply).

Reliability is achieved through the use of unique transaction identifiers, acknowledgements and duplicate removal. This is more effective because explicit connection open and close makes extra load on the communication link. There is an optional user-to-user reliability: the WTP user confirms every received message. Also optional is that the last acknowledgement of the transaction may contain out of band information related to the transaction. For example, performance measurements. Concatenation may be used, where applicable, to convey multiple Protocol Data Units in one Service Data Unit of the datagram transport. The message orientation means that the basic unit of interchange is an entire message and not a stream of bytes. The transactions can be invoked at any time when needed. The protocol provides mechanisms to minimize the number of transactions being replayed as the result of duplicate packets. In case of not proper events an abort message is created and the transaction is aborted. The abort message can be sent initiated from the WTP users or initiated from the WTP providers in case of improper behavior. For reliable invoke messages, both success and failure is reported. The Responder sends back the result as the data becomes available.

4 The test generation process

During the development of a conformance test suite to a protocol several questions are arising already at the very beginning [10]. Different concepts issue in different answers. The following decisions have to be met by the test developers:

- What are the test purposes? "To analyze the right and the false behavior" theoretical purpose has to be translated into unambiguous, well-located concrete tasks. Bigger tests or "micro test cases"?
- What are the formal limits of the test cases? How to choose them, how to match them to the original test purposes? Each test case corresponds to one test purpose? Long test cases? Shorter ones?
- How to guarantee the satisfactory coverage?
- How to collect these tests into groups to support the efficient searching and overview? Storing them without any ordering in stack? Do we check the reaction to not proper events with test cases called invalid and inopportune test cases?

Of course there are the project limits: time, human and technical resources.

Even a small difference of the answers at any of these points results in a completely different test suite. As there are no central rules to have standardized tests, the test suites even for the same protocol can be completely different at different manufacturers. Later, during the interoperability tests, when the co-operation of different implementations is checked, several difficulties may come up. If there were any test or test generation procedure developed showing good performance in these fields and being recommended by a central organization, all market actors would benefit from it, and the R&D, installation and supervision costs would reduce significantly.

4.1 Computer Aided Test Generation

For some problems CATG [2] could give a solution. The time and human resource need seems to fall down to zero, and theoretically very high coverage can be reached "without work".

The experience shows a more pessimistic picture of the CATG concept [13][6]. First it can not miss the developers' active participation: the input of any test generator program has to be a formal unambiguous specification implemented in any computer language that can be processed by the program. This takes much time, and the time gain reached by CATG can not be as great as it seems to be at first sight. Moreover, creating input specification for a program needs much effort and attention, any small mistakes may lead to program errors or erroneous work. It is very hard to analyze codes generated using CATG, so the faults usually turn out only later in real life usage.

The bigger problem is the question of selecting the algorithm to be used. The NP-complete problem of state space exploration of a Communicating Extended Finite State Machine (CEFSM) makes it impossible to completely solve the computer aided test generation. The number of the generated test cases is not linearly proportional to the coverage level, as it gets close to a high percentage, and the ratio of pointless test cases increases. State space exploration algorithms derive the test cases, therefore, depending on the algorithm, two neighbor test cases may come from totally different areas of the state space. They have different length, there are no formal rules, many test cases are parts of other ones, etc.

And to top it all, the whole test suite is in an unstructured stack, and the overview or the execution of subsets according to separable protocol functions is almost impossible. There is also no solution for the proper and reasonable handling of inopportune and invalid test cases.

4.2 Atomic - event pair - test case generation through Test Suite Structure and Test Purposes

Both the traditional and the computer aided ways of test creation have strengths and weaknesses. The traditional method has the advantage of a limited set of proper and easily executable test suite, the CATG methods shorten the development time, etc. We developed a method that takes the advantages and avoids the disadvantages of both older ones [11]. The goal was to develop a systematic way of test generation that has at least average performance in every test suite properties. In order of the points the answers in the test generation process are:

- Each test case has to verify one transition of the SDL diagram. Often, an event pair can be observed by the tester, an input and an output, so we call a single, atomic transition test case an event pair test case.
- The formal appearance and the derivation algorithm have to be common for all test cases. Each test purpose corresponds to one test case, and describes all the necessary information.
- Systematically exploring the whole SDL diagram all the "transition elements", all the event pair test cases can be created. From these elements satisfactory big test suite size can be built up, so the coverage level can meet high requirements. There are no senseless test cases, and no redundancy.
- We collect these test cases during the derivation process into test groups. The test groups identify separable functionality. The groups and the naming conventions support the easy overview and maintenance. We also create invalid and inopportune tests.

We start from a CEFSM, e.g. an SDL specification, which has to contain all the possible events and the whole automata. In contrast to the CATG method in this case the SDL diagram can have informal parts, the aim is to contain the "driver information"- the information needed to understand the functioning of the system.

To develop the whole test, the Abstract Test Suite (ATS) for a system, instead of the traditional, "directly-from-the-standard" method there are three smaller basic steps to be made. The first is to identify the test groups of the Test Suite Structure (TSS). The second is to write down the actual test cases concentrating on the test purposes - to completely define the TSS. We found a systematic procedure to derive the test cases. The last task is to implement the test cases (C code or TTCN tables), and it is almost automatic. Summary: this method consists of smaller, well localized and described tasks automated in many terms. It can be carried out with more efficiency and with less human competence, working experience and protocol behavior information. There are rules for the derivation process and formalisms that make the standardization of this method possible.

4.3 Identifying test groups

When defining the outline of the structure of the system we divide the test cases into basic groups. The question is whether a protocol parameter or variable is a test group identifier. The parameters that are set at the beginning of the test and do not change value during the test (e.g. the class identifier in a test of a classX transaction of the WTP) can be test group identifiers. During the execution of the tests, test groups are usually run together to exhaustively test a given functionality. This is the reason to keep them in a common group.

The basic groups differ the communicating parts (e.g. Client - Server, Initiator - Responder). These groups contain the test of the parts separately, once only tests from one group have to be executed.

The different service groups are the next subgroup in the test suite. Further subgroups are formed based on other relevant features, e.g. optional services.

There are always four different standard test groups at the leaves of the test group tree of the test suite structure hierarchy.

- CA - Capability Tests. The test subgroup provides limited test of the major capabilities of the Implementation Under Test (IUT) aiming to assure that the claimed capabilities are correctly supported, in accordance with the Protocol Implementation Conformance Statement.
- BV - Valid behavior tests. The subgroup verifies that the IUT reacts in conformity with the standard, on receipt or exchange of a valid Protocol Data Unit (PDU). Valid PDU means that the exchange of messages and the content of the exchanged messages are considered as valid.
- BI - Invalid behavior tests. The subgroup verifies that the IUT is in conformity with the standard, on receipt of a syntactically invalid PDU.
- BO - Inopportune behavior tests. The test subgroup verifies that the IUT is capable of a valid reaction, when an inopportune protocol event occurs. Such an event is syntactically correct but it occurs when it is not expected.

Presentation of this step of the procedure on the WTP:

The basic groups distinguish the Initiator from the Responder side. The different transaction classes are the next subgroups (there are three services in WTP, the Class 0, 1 and 2 transaction classes). User acknowledgement is another distinct feature in the WTP. This feature is mandatory and is very important for the user applications. This is why we chose the user acknowledgement as a subgroup of our system. The test group tree of the WTP is shown in Figure 2.

4.4 Defining the Test Suite Structure

After completing the test groups the next step is to create test cases and to add them to the corresponding group. While defining the test purposes, the essential information has to be extracted from the specification. The transitions consist of four main parts:

1. Event - specifying the incoming signals.
2. Condition - that must hold to execute the Action - can also be zero Condition.
3. Action - this is to be done if both the appropriate signal arrives and the Condition holds. Tasks and output signals.
4. Next State - this is the state in which the transition leads the system.

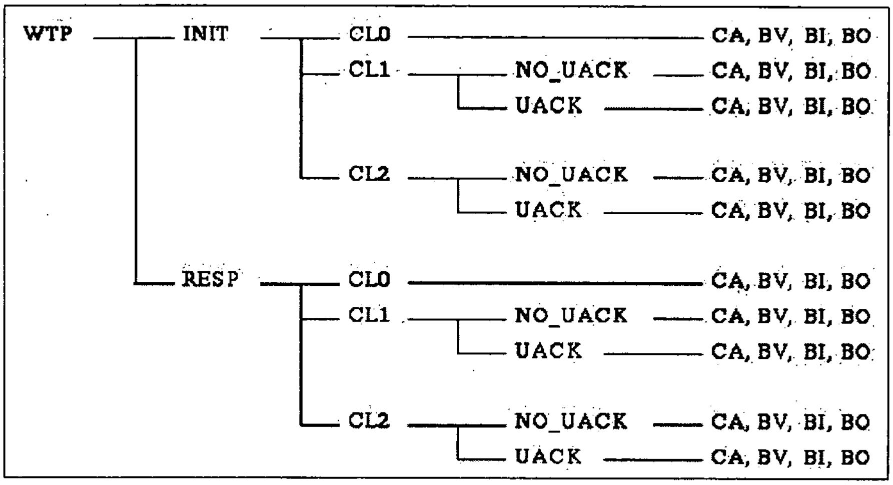


Figure 2: Test group structure of WAP WTP

When specifying test cases we set the following rules:

1. Each "normal" entry corresponds to a single test case. We call an entry a "normal" entry if it contains a single signal and no Condition.
2. Each entry with multiple Conditions corresponds to as many test cases, as many branches of the Conditions are present.
3. Each compound entry (which is for invalid and inopportune signals) corresponds to the following number of test cases:

$$Number\_of\_cases \leq \sum_{i=1}^{num\_sig} \{Sig\_errr\_fields_{sig_i}\}$$

where:

- Number_of_case: number of test cases to make,
- SIG_err_fields: number of signal fields that can contain invalid or inconsistent values,
- num_sig: number of signals that the system can normally receive during normal operation.

We systematically explore the SDL diagram. We start from the "root", the state the automaton enters after start. We describe the test cases that belong to this state, based on the branches of the SDL diagram. We create the formal Test Purpose representations (TP) and place them in the corresponding test group. After completing the test cases for one state, we go to one of the states of the next state level in the hierarchical order of the SDL diagram, and repeat the procedure. All the test cases starting from this new state get the path of a valid test case - starting from the root and resulting in this new state - as preamble. This preamble sets the proper state of the automaton for the test case at the actual testing.

The TP in the TSS&TP document have a strictly defined appearance form. The following Table 1 pattern shows what information the TP provide about the given test cases.

TP Group TP Id	Reference Initial condition Stimulus Expected Behavior

Table 1: Test Purpose representation definition rules

The fields have the following meanings:

- TP Group: This shows the directory structure of the group to which the test case belongs.
- TP Id:. The TP Id is a unique identifier of the test case that is specified according to naming conventions defined in the sub clause below.
- Reference: The reference should contain the references of the subject to be validated by the actual test case (specification reference, clause, paragraph).
- Initial condition: The condition defines in which initial state the IUT has to be to apply the actual test case.
- Stimulus: The stimulus defines the test event to which the test case is related.
- Expected Behavior: The expected behavior is the definition of the events that are expected from the IUT to conform to the base specification. This has to be verified by the test.

We use naming conventions in the TP definitions. The following line shows the rule of names:

Identifier: TP <fm>[<fm>...] x-<nnn>
where

TP:	Test Purpose
<fm>:	functional module
x:	type of testing (CA, BV, BO, BI)

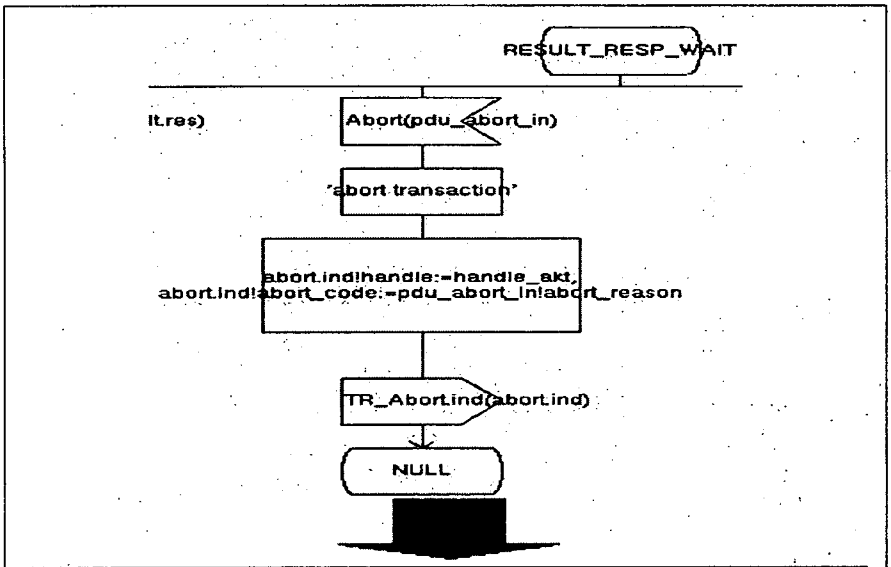
With the help of these formula and rules we manage to make a TSS&TP document where each test purpose representation alone is also capable of telling the user immediately which part of the specification is verified by the given test case.

Presentation of this procedure step on the WTP: For the WTP we had the following test purpose naming conventions for the functional modules:

N	INITIATOR	C	CA, CAPABILITY TESTS
R	RESPONDER	V	BV, VALID BEHAV. TESTS
0	CLASS 0	O	BO,INOPORTUNE BEHAV. TESTS
1	CLASS 1	I	BI, INVALID BEHAV. TESTS
2	CLASS 2		
U	USER ACK		
E	NO USER ACK		

For example the identifier name TPR1EV-006 means: test purpose on the Responder side, Class1, no user acknowledgement, valid behavior test, and this test purpose is the 6th in the group. In Figure 3. the SDL and the test purpose representation can be seen.

The complete TSS&TP document defines the test cases and the test information, the whole dynamic behavior of the test. The test code can be easily created with the help of this document.



WTP//INIT/CL2/NO_UACK/ TPN2EV-006	**Initial condition:** IUT has entered the RESULT_RESP_WAIT state. **Stimulus:** An Abort PDU arrives. **Verify** that the IUT enters the NULL state and sends a TR_Abort.ind ASP.

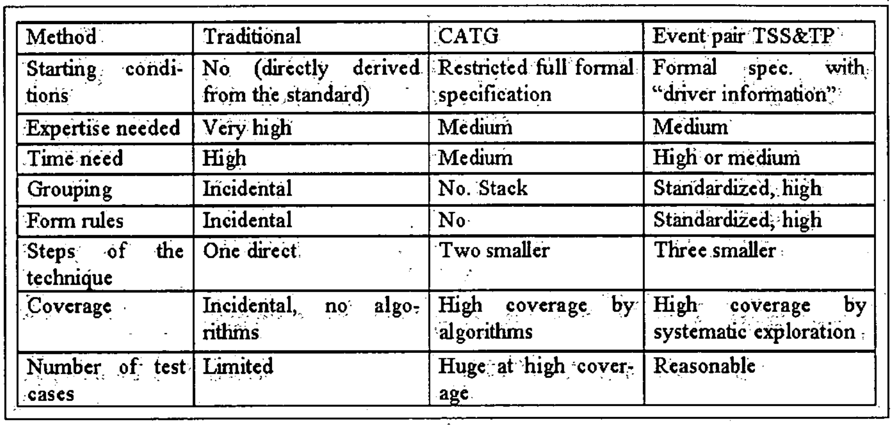
Figure 3: Derivation of a TP for a test case

4.5 Comparison to the traditional and CATG methods

Today's protocol specifications are usually based on the EFSM semantics. With the introduction of variables state space explosion may occur. So, the state space of an EFSM model can be extremely large and even infinite. Theoritically, the whole state space has to be explored and on each possible state transition a decision has to be made, if it is to be included in the test suite. This is a complex problem, that is hard to formalize. Thus, the selection of a limited appropriate set of traces – i.e. the test cases – using CATG methods, which are based on the current technology and theoritical background, fails to be useful in practice.

Our experience shows the same, the different CATG techniques resulted in an unstructured set of inefficient test cases on WAP WTP and other protocols[11]. After checking our method on these protocols and reading additional reports on CATG, we concluded the performance factors presented in Figure 4 [6][5]. The main reason for this result is easy to explain. Many parts of the test generation process can be automatized. Nevertheless, beyond a certain limit human intelligence cannot yet be substituted by pure computer based solutions. Our method tries to utilize systematic standard steps, but retains human intelligence for meeting complex decisions.

According to the previous statements, we found that our method is more practical than the traditional ones, if there is lack of highly experienced developers and the high quality of the test suite is a requirement. The Event pair TSS&TP method provides better overview, execution and easier maintenance than the CATG methods.



Method	Traditional	CATG	Event pair TSS&TP
Starting conditions	No (directly derived from the standard)	Restricted full formal specification	Formal spec. with "driver information"
Expertise needed	Very high	Medium	Medium
Time need	High	Medium	High or medium
Grouping	Incidental	No. Stack	Standardized, high
Form rules	Incidental	No	Standardized, high
Steps of the technique	One direct	Two smaller	Three smaller
Coverage	Incidental, no algorithms	High coverage by algorithms	High coverage by systematic exploration
Number of test cases	Limited	Huge at high coverage	Reasonable

Figure 4: Comparison of test generation methods

5 The checking of the model of the WAP WTP standard

In the frames of this paper we also want to note that during our work, at the verification phase [3][7] preceding the test generation process, we checked the model of the WAP WTP. Interestingly, we found flaws in the specification, which could possibly cause problems in the operation of the protocol. We validated [4] and simulated the SDL system to examine the protocol specification itself and two shortcomings arose.

5.1 Result waiting feature that may cause a problem

The first problem arises during a Class 2 transaction, if the next higher layer of the protocol on the Responder side stops its operation in a certain transaction period. In this situation both the WTP Responder and the WTP Initiator are in the state called RESULT WAIT, according to the standard. The WTP Responder is waiting for the corresponding incoming abstract service primitive (ASP) signal. If it does not come and there are no timers running, both communicating parts could wait forever for an event - that is an error on the Responder side, which inhibits the functioning of the Initiator. This is practically a deadlock, which is not resolved in this layer according to the specification. Right now the only solution could be to implement some kind of a timer in the next higher layer over the Initiator, which starts when this situation could possibly happen, and that sends an ASP signal to the Initiator after a long time without response. The first message sequent chart shows the critical situation in Figure 5.

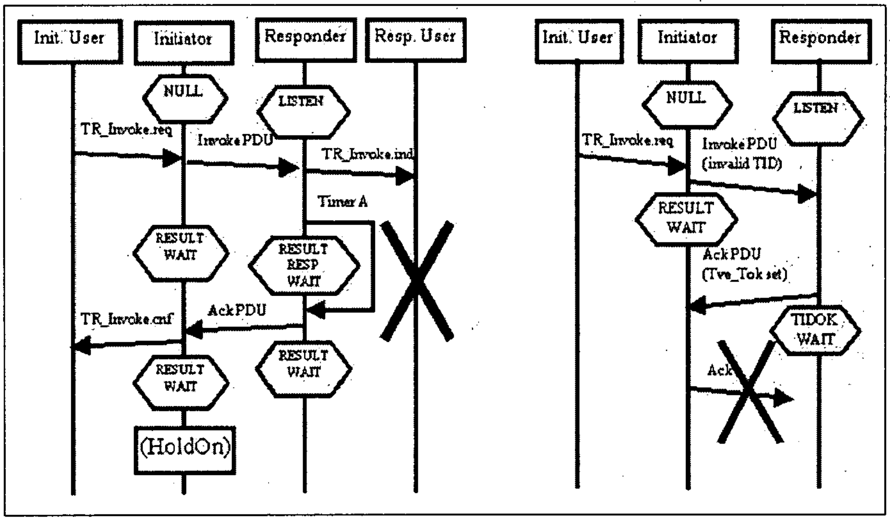


Figure 5: Critical situations in the WAP WTP model

5.2 The transaction identifier verification problem

The second problem can arise during either a Class 1 or a Class 2 transaction. The transaction identifier verification process ensures the proper sequence of the WTP PDUs. If this function has to run, and the connection between the Initiator and the Responder temporarily fails, the Responder can get stuck in the state called TIDOK WAIT. The only way to move the process out of this state is with the help of the WTP Initiator, which has to send a particular PDU signal. If this does not arrive to the Responder than it gets stuck. There is no timer that could move the process out of this state, and the next higher layer does not even get a notice about the state of the process, so a timer can not be implemented either. Even if the connection is restored, the process remains in this state. This situation can also arise, if a bad or intentionally modified Initiator implementation does not perform the right functioning. A malice communicating part can open several transactions, and leave them in deadlock in the Responder side. The second message sequent chart shows the critical situation in Figure 5.

6 Conclusion

Our test generation method consists of smaller well localized and described tasks, automated in many terms. It can be carried out with more efficiency and less human competence, working experience and protocol behavior information. There are defined formalisms and rules for the derivation process. With the help of the TSS&TP document it is now possible to test implementations of the protocol without knowing the protocol itself. It enables generating fully formal test step descriptions (for example TTCN tables) almost automatically. This test suite is more standardized, can provide high coverage, and has good or at least average performance in almost every problematic field of the test generation process.

During our work we found some flaws in the WAP WTP protocol specification, which could - under special circumstances - possibly cause problems in the operation of the protocol.

References

[1] Gregor V. Bochmann and Alexandre Petrenko. Protocol testing: review of methods and relevance for software testing. *International Symposium on Software Testing and Analysis*, pages 109 – 124, 1994.

[2] C. Bourhfir, R. Dssouli, and E. M. Aboulhamid. *Automatic Test Generation for EFSM-based Systems.* http://citeseer.nj.nec.com/114451.html.

[3] EG 201 383 ETSI Guide. Methods for testing and specification (mts); use of sdl in etsi deliverables; guidelines for facilitating validation and the develop-ment of conformance tests, 1999.

[4] TCTR 004 ETSI Technical Committee Technical Report. Methods for testing and specification (mts); reports on experiments in validation methodology. Technical report, ETSI, 1996.

[5] TR 101 051 ETSI Technical Report. Methods for testing and specification (mts); report of the catg applications. Technical report, ETSI, 1999.

[6] TR 101 279 ETSI Technical Report, 1948.

[7] ETR 184 ETSI Technical Review. Methods for testing and specification (mts); over-view of validation techniques for european telecommunication standards (etss) containing sdl. Technical report, ETSI, 1995.

[8] WAP Forum. Wireless application protocol architecture specification, April 1998.

[9] WAP Forum. Wireless application protocol wireless transaction protocol specification, February 2000.

[10] B. Gregor and V. Petrenko. Protocol testing: review of methods and relevance for software testing, 1994.

[11] R. Horváth, Z. Pap, and Z. Réthâti. Methods for telecommunication protocol development and conformance testing. *Student Conference BUTE*, 2000.

[12] ITU-T. *Recommandation Z.100: Specification and Description Language*, 1992.

[13] B. Koch, J. Grabowski, D. Hogrefe, and M. Schmitt. Autolink a tool for automatic test generation from sdl specications, 1998.

[14] Gang Luo, Gregor v. Bochmann, and Alexandre Petrenko. Test selection based on communicating nondeterministic finite-state machines using a generalized Wp-method. *IEEE Transactions on Software Engineering*, 20(2):149–162, February 1994.