MBT for GlobalPlatform Compliance Testing: Experience Report and Lessons Learned

Gil Bernabeu*, Eddie Jaffuel†, Bruno Legeard‡§ and Fabien Peureux‡§
*GlobalPlatform, USA, gil.bernabeu@globalplatform.org
†eConsult, France, eddie.jaffuel@econsult.fr
‡Smartesting R&D Center, France, {legeard,peureux}@smartesting.com
§Institut FEMTO-ST, France, {blegeard,fpeureux}@femto-st.fr

***Abstract*—Compliance testing is done to determine whether a system meets a specified standard prescribed by a given authority. One key goal of compliance testing is to ensure interoperability between systems, on the basis of agreed norms and standards, while also allowing acceptable variations to be inserted by the compliant product vendors. This paper reports on the deployment of a Model-Based Testing approach, based on the Smartesting solution, to produce compliance test suites for GlobalPlatform Specifications, which aim to ensure the long-term interoperability of embedded applications on secure chip technology. After explaining the context and the motivation to use a Model-Based Testing approach as a part of the GlobalPlatform Compliance Program since 2007, the paper describes the GlobalPlatform Working Group testing process, and discusses the lessons learned from this success story.**

***Keywords*-Model-Based Testing, Compliance Test Suite, GlobalPlatform, Operational Experiments Feedback.**

I. Context and Motivation

GlobalPlatform[1] (GP for short) is a cross-industry and not-for-profit association, which members are payment organizations such as American Express, MasterCard, or Visa International, telecom operators, like AT&T, France Telecom, NTT or Verizon and industrial leaders (AMD, Apple, Blackberry, Gemalto, Nokia, Samsung, etc.). GlobalPlatform identifies, develops and publishes specifications facilitating secure and interoperable deployment and management of multiple embedded applications on secure chip technology. Its proven technical specifications are regarded as the international industry standard for building a trusted end-to-end solution serving multiple actors and supporting several business models. The freely available specifications provide the foundation for market convergence and innovative new cross-sector partnerships. As depicted in Fig. 1, the technology has been adopted globally across finance, mobile/telecom, government, healthcare, retail and transit sectors. GlobalPlatform also supports an open compliance program ecosystem to ensure the long-term interoperability of secure chip technology. Recent research conducted by Eurosmart confirmed that 2012 shipments of microcontroller smart secure devices (secure chips) are over 7 billion units, of which 2.6 billion units leverage GlobalPlatform technology.

[1]http://www.globalplatform.org

GlobalPlatform is the standard for managing applications on secure chip technology
TRUSTED EXECUTION ENVIRONMENT
MESSAGING
SECURE ELEMENT
Across several market sectors and in converging sectors
PREMIUM CONTENT
FINANCIAL
TELECOM
GOVERNMENT
AUTOMOTIVE
HEALTHCARE
RETAIL
TRANSIT

Fig. 1. GlobalPlatform Business Ecosystem

A. Historical GP Compliance Program Process

Key to market stability is the adoption of proven standards. Industry acceptance of any standard or specification, however, will only be achieved if there is a commitment to maintain the stability of the specifications to encourage their adoption. Standards, specifications and configurations are only as dependable as the products developed to abide by them. This is why it is important that the industry has a means to test and verify product compliance. To achieve this, the first GP Compliance Program, created in 2001, was organized using a classical process:

1) One test tool vendor was selected to create a test suite.
2) The selected test tool vendor created the test suite.
3) The GP Compliance Working Group reviewed and validated the given test suite.
4) Once committed, this test suite was used (using the testing tool of the selected tool vendor) to qualify the products.

However, this initiative was stopped because (i) the ecosystem had little interest in validating the provided test suite, (ii) each product vendor was using an in-house test environment or had established a specific contract with another test tool vendor, and (iii) the scope of testing was unclear. To overcome these obstacles, the GP compliance process was advanced to meet the requirements of its users.

B. Current GP Compliance Program Process

Since 2007, the GP Compliance Program is managed using a unified process, which is described in Fig 2. The Specification Working Group is in charge to define the specifications and the configurations. The Compliance Working Group reviews the coverage of the compliance test suite, and arbitrates interpretation of the specification when necessary. The Compliance Secretariat manages the test suite creation and maintenance, and organizes the *TestFests*.

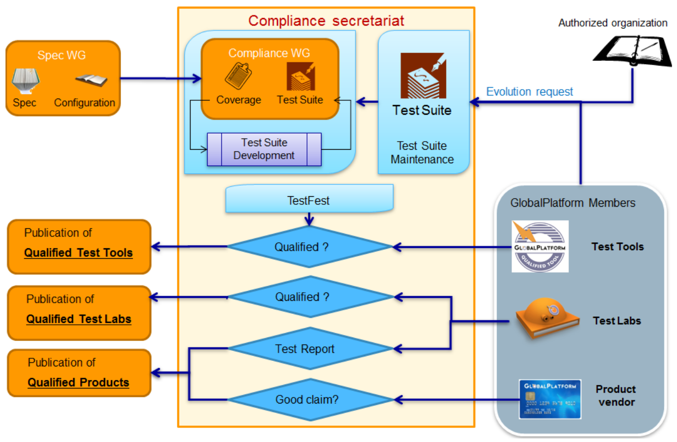


Fig. 2. Process of the GlobalPlatform Compliance Program

The TestFests comprise 3 or 4 days of face-to-face meetings, involving the GP Secretariat, the test tool providers (usually 3 to 5 companies) and the product vendors (usually 2 to 4 companies). The ultimate goal of a TestFest is to qualify the test tools (software and test harness), regarding a given compliance test suite. During a TestFest, all tests of the test suite that remain unchanged throughout the TestFest (even if some test cases may be excluded) are executed on every test tool and tests that are considered as wrong are excluded. The product being tested remains unchanged during the TestFest: only the test tool providers may correct their software or test harness during the TestFest when expected results are marked as wrong. At the end of the TestFest, each test tool must produce the same result as the expected result for each product, otherwise the test tool will not be qualified. The GlobalPlatform compliance ecosystem is therefore equipped with qualified test tools and qualified test laboratories. To ensure the success of these events, which ensure the delivery of an efficient GP Compliance Program, the next subsection introduces the two major challenges, regarding the test suite development that needs to be addressed.

C. Challenges about Test Suite Development

There were two major challenges for the GlobalPlatform Compliance Program that had to be addressed regarding compliance test suite development.

1) The test suite shall be open to any test execution tools: In a classical document-based approach, the test plan is described in an informal document, providing the initial conditions and the test sequences at a high level (focusing on functionality instead of technical details). Different test tools providers will implement the corresponding test scripts, which are detailed enough for execution. These implementations are subject to interpretation because the test plan documentation is informal and at a high level, and as such is ambiguous and, at times, incomplete by nature.

2) The variants in specification and in test suite shall be managed efficiently: GlobalPlatform is managing generic specifications along side various specific configurations, which are derived from the generic one. Each configuration is defined for a specific domain or product and meets different the market needs. GP then develops a compliance test suite for each specific configuration (and not for the generic specification). By definition, the generic specification is very open and forms the basis for many market-focused configurations to be developed. In a generic specification, the features are often optional (using the word *MAY* in the text), alternative options are described, and sometimes the generic specification leaves blank some parts or indicates them as implementation-dependent. In contrast, a specific configuration limits the scope of the generic specification. Even if the specific configuration is mostly referring to the generic specification, the specific configuration clarifies the blank points, indicates which options have to be supported (using the word *MUST* or *MUST NOT* in the text). Knowing that the purpose is to have a compliance test suite for each specific configuration, the challenge is to understand which parts of the test suite can be reused in order to avoid duplicated effort.

D. Model-Based Testing Integration

To tackle these issues, the GlobalPlatform consortium has decided since 2007 to use a Model-Based Testing approach [1] using the tool *CertifyIt* provided by the company Smartesting[2]. Basically, Model-Based Testing [2] (MBT for short) refers to the processes and techniques for the automatic derivation, from formal models, of abstract test cases (abstract because it relies on the models). It allows the generation of concrete tests from abstract tests, and the manual or automated execution of these concrete tests. The Smartesting MBT solution takes as input a UML model of the expected behaviour of the system under test, and uses model coverage criteria to automatically generate test cases, which can be translated into executable test scripts.

Section II describes the test generation process, based on the Smartesting solution, to produce the compliance test suites, whereas Sect. III and IV respectively gives the current results regarding the GP compliance test suite generation, and discusses the lessons learned from this experience. Finally, Sect. V concludes the paper by giving the major feedback and the current status of this success story.

[2]http://www.smartesting.com

II. Model-based Compliance Test Generation

Figure 3 gives an overview of the MBT process to address the GlobalPlatform conformance issues. The process starts on the left at the textual requirements, from which a test designer team derives the Test Objective Charter and a UML test model, enabling the automated generation of abstract compliance test suites. Finally, an adaptation layer generates compliance test scripts, which can be directly executed on a system by a manual tester. These steps are described in the next sections.

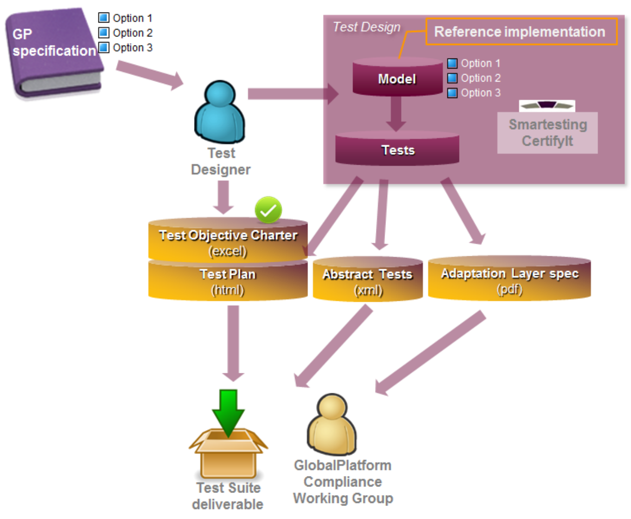


Fig. 3. Process of the Model-Based Compliance Test Suite Generation

A. *From GP Specifications to Test Objective Charter*

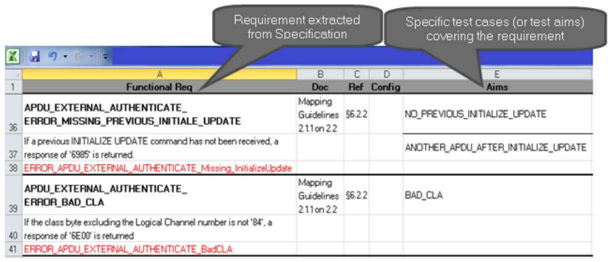
Basically, a Test Objective Charter (TOC) takes the form a an Excel sheet (as shown in Fig. 4), which is composed of:

- A *Requirement List*, which is textual extracts from the specifications with requirement identifiers in order to produce a clear traceability link between the tests and the specifications at the end.
- For each requirement, a list of *Test Aims* (also called *test objectives* or *test cases*) and *Expected Observations* (also called *pass criteria* or *verifications*).

Such a TOC does not contain the prerequisites (like required personalizations), neither the test sequences details. Indeed the purpose of a TOC is to discuss and get an agreement on the coverage (the *WHAT to cover*) before starting the entire test suite definition (the *HOW to cover*). For the GlobalPlatform purpose, the TOC usually relies on one specific configuration.

B. *From GP Specifications and TOC to Test Model*

The test model represents the expected behavior of the Application Protocol Data Unit (APDU) specified in the GlobalPlatform standard. Within the Smartesting approach, a subset of UML, called UML4MBT [3], is used: it includes UML class diagrams, state machines and OCL constraints to formalize the control points and observation points, the expected dynamic behavior and business entities described in the standard [4].


Requirement extracted from Specification
Specific test cases (or test aims) covering the requirement

	A	B	C	D	E
1	Functional Req	Doc	Ref	Config	Aims
36	APDU_EXTERNAL_AUTHENTICATE_ ERROR_MISSING_PREVIOUS_INITIALE_UPDATE	Mapping Guidelines 2.1.1 on 2.2	§6.2.2		NO_PREVIOUS_INITIALIZE_UPDATE
37	If a previous INITIALIZE UPDATE command has not been received, a response of '6985' is returned.				ANOTHER_APDU_AFTER_INITIALIZE_UPDATE
38	ERROR_APDU_EXTERNAL_AUTHENTICATE_Missing_InitializeUpdate				
39	APDU_EXTERNAL_AUTHENTICATE_ ERROR_BAD_CLA	Mapping Guidelines 2.1.1 on 2.2	§6.2.2		BAD_CLA
40	If the class byte excluding the Logical Channel number is not '84', a response of '6E00' is returned				
41	ERROR_APDU_EXTERNAL_AUTHENTICATE_BadCLA				

Fig. 4. Excerpt of a Test Objective Charter File

IBM Rational Software Architect with Smartesting CertifyIt testing plug-in is used to design the test model dedicated for the testing of the specifications (including generic specification and specific configurations), but restricted to the Test Objective Charter. The test model also contains documentation on UML element (like method or enumeration literal). This documentation helps to make the bridge between the abstraction level of the model, the concrete command and data that have to be executed on the concrete application (it will be part of the generated Adaptation Layer Specification introduced later). This test model encompasses the options available by the generic specification, and describes the behavior of the system if the option is supported or not. The test model has a dedicated architecture (see Fig 5) that allows managing the common part of the generic specification and also the specific part. The generic common model part is reused (as an inclusion) by the specific model part.

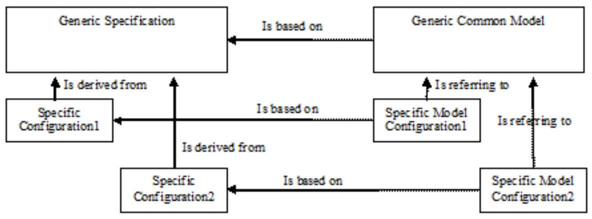
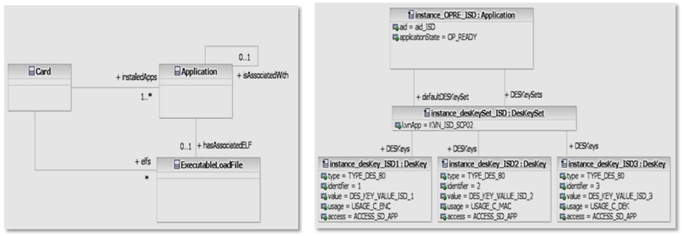


Fig. 5. Test Model Architecture

Figures 6a and 6b respectively introduce an example of class diagram (depicting executable load files that can be executed by some applications embedded in a card) and an example of object diagram that instantiates this kind of architecture.



(a) Class Diagram (b) Object Diagram

Fig. 6. Examples of UML Test Model Diagram

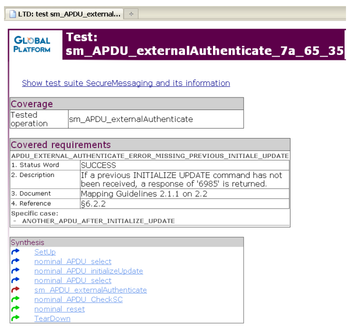
C. *Generating the Test Suite from the Test Model*

Such a test model is precise and complete enough to allow automated derivation of the compliance test suite. This derivation is a fully automated process supported by the Smartesting CertifyIt testing tool [5], which generates abstract test cases (abstract because relying on the UML test model) to cover the items of the Test Objective Charter file. Each generated test case is typically a sequence of APDUs, with input parameters and expected output values for each action. More precisely, three test suite artifacts are generated and derived from the test model: (i) the test suite in XML format for test execution purpose, (ii) the test suite in HTML for test documentation purpose and (iii) the Adaptation Layer specification. The next subsections give an overview of these generated artifacts.

D. *Publishing the Test Suite Artifacts*

1) Test Suite in XML and HTML: The test suite includes the following:

- The abstract test cases (or abstract test sequences), which gives the necessary steps to fulfil the initial conditions of the tests, and also the test sequence covering the test objective and checking the pass criteria. The abstracts test cases describe all the commands with input data and expected output data. The abstract test cases are published into two formats: XML for execution purpose (it is to be imported into test execution tools) and in HTML for documentation purpose (it is to be used for human review and understanding). Figures 7a and 7b respectively show an example of a test case presentation in HTML format, in which the sequence of APDU calls is described in the *Synthesis* frame at the bottom, and an excerpt of the details of each APDU call with the input and expected data (the figure precisely shows the two calls *nominal_APDU_select* and *nominal_APDU_initializeUpdate*, which are the second and the third steps of the test case introduced in Fig. 7a).



LTD: test sm_APDU_external...

GLOBAL PLATFORM **Test: sm_APDU_externalAuthenticate_7a_65_35**

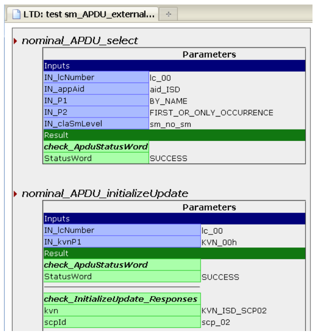
Show test suite SecureMessaging and its information

Coverage	
Tested operation	sm_APDU_externalAuthenticate

Covered requirements	
APDU_EXTERNAL_AUTHENTICATE_ERROR_MISSING_PREVIOUS_INITIALE_UPDATE	
1. Status Word	SUCCESS
2. Description	If a previous INITIALIZE UPDATE command has not been received, a response of '6985' is returned.
3. Document	Mapping Guidelines 2.1.1 on 2.2
4. Reference	§6.2.2
Specific case: - ANOTHER_APDU_AFTER_INITIALIZE_UPDATE	

Synthesis
- SetUp
- nominal_APDU_select
- nominal_APDU_initializeUpdate
- nominal_APDU_select
- sm_APDU_externalAuthenticate
- nominal_APDU_CheckSC
- nominal_reset
- TearDown

(a) Test Case



LTD: test sm_APDU_external...

nominal_APDU_select

Parameters	
Inputs	
IN_lcNumber	lc_00
IN_appAid	aid_ISD
IN_P1	BY_NAME
IN_P2	FIRST_OR_ONLY_OCCURRENCE
IN_claSmLevel	sm_no_sm
Result	
check_ApduStatusWord	
StatusWord	SUCCESS

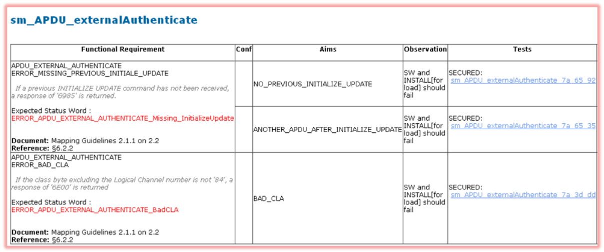
nominal_APDU_initializeUpdate

Parameters	
Inputs	
IN_lcNumber	lc_00
IN_kvnP1	KVN_00h
Result	
check_ApduStatusWord	
StatusWord	SUCCESS
check_InitializeUpdate_Responses	
kvn	KVN_ISD_SCP02
scpId	scp_02

(b) Test Case Steps

Fig. 7. Examples of Test Suite Document in HTML Format

- The Test Objective Charter information (introduced at the beginning of this section) including now the link from the requirements to the abstract test cases through a traceability matrix. Figure 8 shows an example of such a traceability matrix in HTML format.

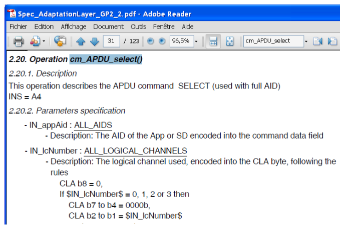


sm_APDU_externalAuthenticate

Functional Requirement	Conf	Aims	Observation	Tests
APDU_EXTERNAL_AUTHENTICATE ERROR_MISSING_PREVIOUS_INITIALE_UPDATE *If a previous INITIALIZE UPDATE command has not been received, a response of '6985' is returned.* Expected Status Word : ERROR_APDU_EXTERNAL_AUTHENTICATE_Missing_InitializeUpdate **Document:** Mapping Guidelines 2.1.1 on 2.2 **Reference:** §6.2.2		NO_PREVIOUS_INITIALIZE_UPDATE	SW and INSTALL[for load] should fail	SECURED: sm_APDU_externalAuthenticate_7a_65_92
		ANOTHER_APDU_AFTER_INITIALIZE_UPDATE	SW and INSTALL[for load] should fail	SECURED: sm_APDU_externalAuthenticate_7a_65_35
APDU_EXTERNAL_AUTHENTICATE ERROR_BAD_CLA *If the class byte excluding the Logical Channel number is not '84', a response of '6E00' is returned* Expected Status Word : ERROR_APDU_EXTERNAL_AUTHENTICATE_BadCLA **Document:** Mapping Guidelines 2.1.1 on 2.2 **Reference:** §6.2.2		BAD_CLA	SW and INSTALL[for load] should fail	SECURED: sm_APDU_externalAuthenticate_7a_3d_dd

Fig. 8. Traceability Matrix from TOC Items to Test Cases Description

2) Adaptation Layer Specification: The Adaptation Layer (AL for short, and also called Automation Layer or Abstraction Layer) implements the mapping between the abstract keywords used in the test model and the low level commands to be executed on the system under test. Hence, the Adaptation Layer specification is automatically generated from the test model (and the documentation attached to the modeling structures), while its implementation is dependent on the test execution tool environment and language, and so is left in charge of the test tool providers. As shown in Fig. 9, the generated files allow the user to document the APDU commands Fig. 9a as well as the concrete data to be used (Fig. 9b) in order to concretize and execute the generated test suite.



Spec_AdaptationLayer_GP2_2.pdf - Adobe Reader
Fichier Edition Affichage Document Outils Fenêtre Aide
31 / 123 96,5% cm_APDU_select

2.20. Operation cm_APDU_select()

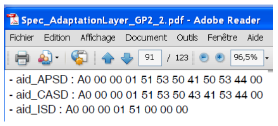
2.20.1. Description

This operation describes the APDU command SELECT (used with full AID)
INS = A4

2.20.2. Parameters specification

- IN_appAid : ALL_AIDS
  - Description: The AID of the App or SD encoded into the command data field
- IN_lcNumber : ALL_LOGICAL_CHANNELS
  - Description: The logical channel used, encoded into the CLA byte, following the rules
    CLA b8 = 0,
    If $IN_lcNumber$ = 0, 1, 2 or 3 then
      CLA b7 to b4 = 0000b,
      CLA b2 to b1 = $IN_lcNumber$

(a) APDU Command



Spec_AdaptationLayer_GP2_2.pdf - Adobe Reader
Fichier Edition Affichage Document Outils Fenêtre Aide
91 / 123 96,5%

- aid_APSD : A0 00 00 01 51 53 50 41 50 53 44 00
- aid_CASD : A0 00 00 01 51 53 50 43 41 53 44 00
- aid_ISD : A0 00 00 01 51 00 00 00

(b) Constant Data

Fig. 9. Extracts of Adaptation Layer Specifications

III. Technical Results

From 2007 till present, the GP Compliance Program has been using this MBT process to produce its compliance test suite. The metrics of this GP Compliance Program in summer 2014 are the following: 15 active compliance test suites have been generated using the current model-based compliance test suite development (i.e. excluding the previous versions of the test suites, which were manually designed), comprising in nearly 6000 tests cases. These test suites are used every year during the TestFests to qualify the test tools, which are then used by approved GlobalPlatform laboratories. Their relevance constitutes a crucial and indispensable key factor that makes these events successful.

Technically, the developed compliance test suites cover the following configurations, derived from the generic specification GlobalPlatform Card Specifications version 2.2.1 [6]:

- UICC Configuration v1.0.1 (Telecom configuration)
- Contactless Extension for UICC v1.0 (Telecom configuration)

- Mapping Guidelines v1.0.1 (Banking configuration)
- Basic Financial Configuration (Banking configuration)
- ID Configuration v1.0 (Identity Configuration)
- Common Implementation Requirements v1.0 (Generic Configuration)
- Secure Element Configuration v1.0 (Embedded Secure Element Configuration)

Several test suites are also related to other independent specifications (without configuration), such as Card Compliance (SWP/HCI test suites from ETSI, Amendment B (SCP81), Secure Element Access Control v1.0 - applet side) and Device Compliance (Secure Element Access Control v1.0 - device side, Open Mobile API Specification v2.05 from SIMalliance consortium - transport API, and Trusted Execution Environment).

This approach has proven to be very successful: extensions of the use of the proposed model-based testing approach have already been conducted and confirm the relevance and benefits of the approach. It mainly concerns modules of the GlobalPlatform System Group. Basically, it is a Trusted Service Manager (TSM) and is a role in a Near Field Communication (NFC) ecosystem. It acts as a neutral broker that sets up business agreements and technical connections with Mobile Network Operators (MNOs), phone manufacturers or other entities controlling the secure element on mobile phones. The TSM enables service providers to distribute and manage their contactless applications remotely by allowing access to the secure element in NFC-enabled handsets.

New integrations are also being conducted, especially regarding the GlobalPlatform Device Group, which defines trusted applications running inside a Trusted Execution Environment (TEE). The TEE is a secure area that resides in the main processor of a smart phone (or any mobile device) and ensures that sensitive data is stored, processed and protected in a trusted environment. The TEE ability to offer safe execution of authorized security software, known as *Trusted Applications*, enables it to provide end-to-end security by enforcing protection, confidentiality, integrity and data access rights.

IV. Lessons Learned From Experience

This section summarizes the lessons learned about MBT integration for GlobalPlatform compliance issues, regarding the valuable benefits noticed from this experience, and according to the initial challenges introduced in Sect. I-C.

A. The test suite shall be open to any test execution tools

Since test cases are provided as abstract in a standard XML format, the following benefits are delivered:

- The test suite is open to any test-execution tool.
- As the abstract test scripts are delivered in XML format, it reduces the effort for the execution tool supplier to implement the test suite, because they are only implementing the complementary part of the Adaptation Layer.
- All the test tools are using the same scripts, which avoid contradictory interpretation of the test documentation.

B. The variants in test suite shall be managed efficiently

The use of model-based approach makes it possible to give structure to the whole process as following:

- The test model is used as a reference and can be reused since it is usually composed of a generic part (specifying the *MUST* and *MUST NOT* of the specification) and a specific part (specifying the *MAY* of the specification).
- The Test Objective Charter, as well as test suites and adaptation layer specification, can also be reused.

C. Other Benefits

Other benefits, directly inherited from well-known advantages of the MBT approaches [7], have been noticed within the MBT compliance process. It mainly concerns:

- Test case generation is an automated process and so more predictive and less error-prone than manual processes.
- It provides the generated test cases with a clear functional coverage metric from the viewpoint of the TOC.
- All generated assets (XML and HTML test suites, Adaptation Layer Specification) are kept in sync because they are derived from one common asset (the test model).
- It helps to leverage existing investments in test management skills, tools, and processes.
- It reduces test maintenance costs because only the test model is managed instead of the test cases.

V. Conclusion and Future Works

From the launch of the GP Compliance Program in 2007 until the present day, GP has been using Model-Based Testing to produce its compliance test suite, which is now providing a unique value: the test suite (i.e. scripts) is usable for integration to the product vendors in-house systems, the test suite is open to any test tool vendor, the test suite is validated in real-time, and the test suite supports product variants. This MBT integration is therefore a concrete success story, which today enables and motivates GP Working Groups to extend its use, which is currently focused on system compliance.

References

[1] G. Bernabeu and N. Lavabre, “Model-based testing for a world-wide compliance program,” in 1^{st} *User Conference on Advanced Automated Testing (UCAAT'13)*, Paris, France, Oct. 2013, avalaible at http://ucaat.etsi.org/2013/presentations/Keynote_MBT%20for%20a%20Compliance%20Program-GlobalPlatform-GilBernabeu.pdf.

[2] M. Utting and B. Legeard, *Practical Model-Based Testing - A tools approach*. San Francisco, CA, USA: Morgan Kaufmann, 2006.

[3] F. Bouquet, C. Grandpierre, B. Legeard, F. Peureux, N. Vacelet, and M. Utting, “A subset of precise UML for model-based testing,” in *Proc. of the 3^{rd} Int. Workshop on Advances in Model-Based Testing (AMOST'07)*. London, UK: ACM Press, Jul. 2007, pp. 95–104.

[4] E. Bernard, F. Bouquet, A. Charbonnier, B. Legeard, F. Peureux, M. Utting, and E. Torreborre, “Model-based testing from UML models,” in *Proc. of the Int. Workshop on Model-Based Testing (MBT'06)*, ser. LNCS, vol. 94. Dresden, Germany: Springer, Oct. 2006, pp. 223–230.

[5] F. Bouquet, C. Grandpierre, B. Legeard, and F. Peureux, “A test generation solution to automate software testing,” in *Proc. of the 3^{rd} Int. Workshop on Automation of Software Test (AST'08)*. Leipzig, Germany: ACM Press, May 2008, pp. 45–48.

[6] *GlobalPlatform Card Specification Version 2.2.1*, Jan. 2001, avalaible at http://www.globalplatform.org/specificationscard.asp.

[7] A. Dias-Neto and G. Travassos, “A Picture from the Model-Based Testing Area: Concepts, Techniques, and Challenges,” *Advances in Computers*, vol. 80, pp. 45–120, Jul. 2010, ISSN: 0065-2458.