Model Based Testing for SCA Conformance Testing

Approach and First Experience

Julien Botella
Eddie Jaffuel
Bruno Legeard
Fabien Peureux

Oct 2014

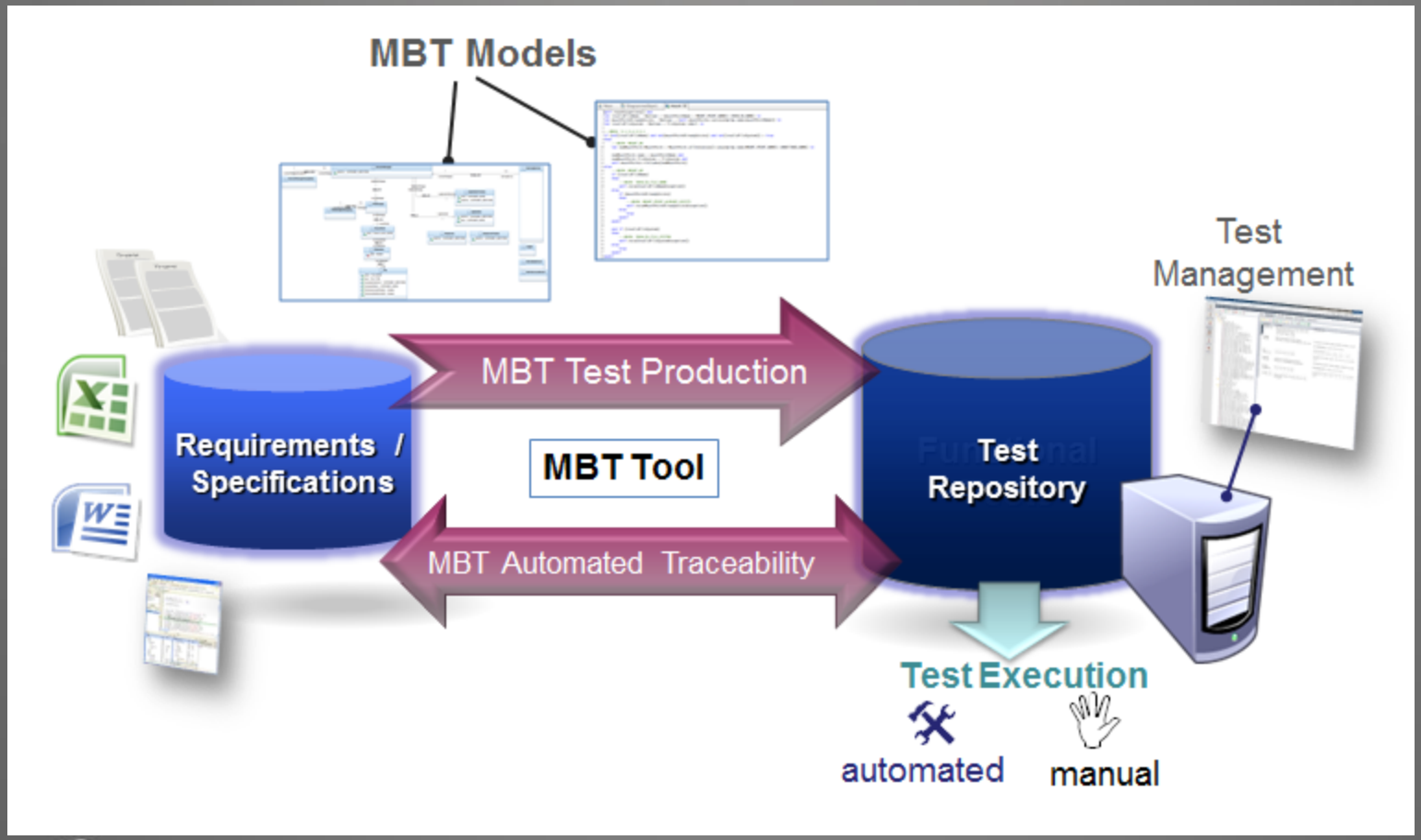
Agenda

- Context and motivations
- MBT for Conformance Testing – GlobalPlatform compliance program
- Application to SCA 2.2.2
- Lessons learned and Conclusion

Agenda

- Context and motivations
  - Introduction of Model-Based Testing
  - Model-Based testing for Conformance Testing
- MBT for Conformance Testing – GlobalPlatform compliance program
- Application to SCA 2.2.2
- Lessons learned and Conclusion

Introduction to Model-Based Testing



MBT for Conformance Testing

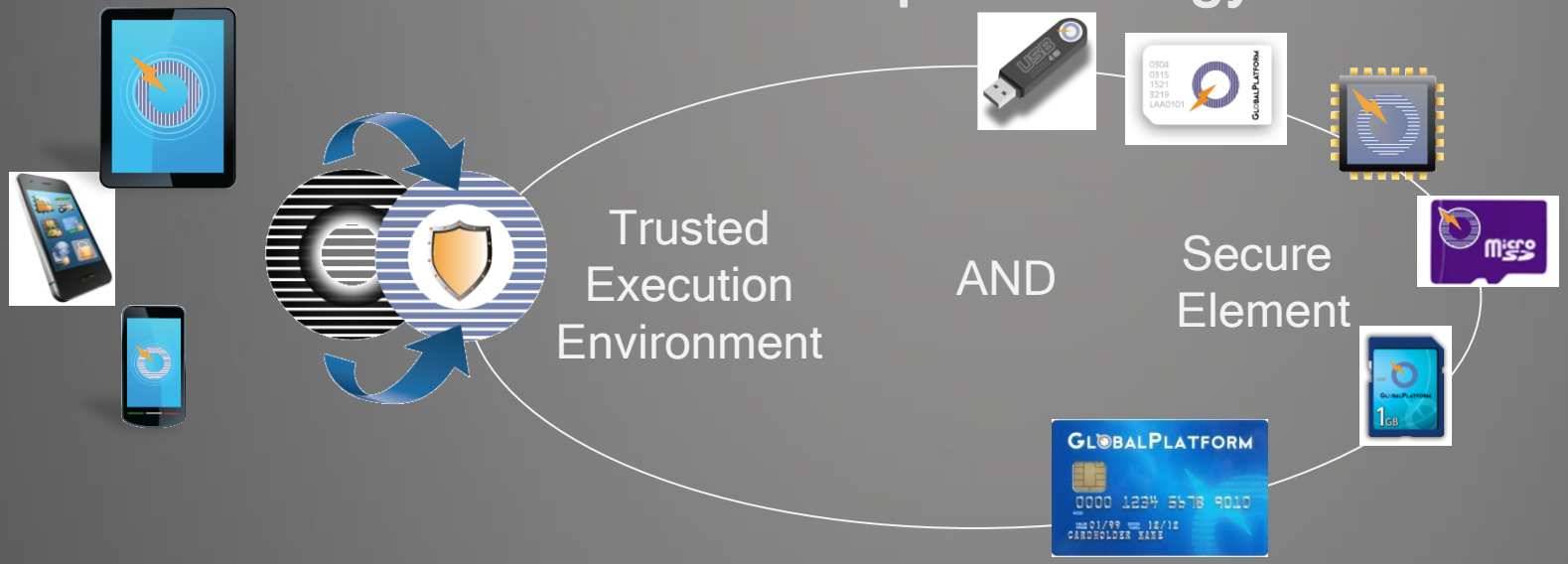
- MBT modeling from a standard (e.g. SCA 2.2.2 sepcifications)
- Test selection criteria are used to generate the adequate conformance test suite
  - Typically, 2 to 5 abstract test cases for each functional requirement
- Automated publication of abstract generated conformance tests:
  - To create a documented test repository (using a test management tool), including traceability between requirements and tests
  - To generate executable test scripts for test execution automation

Agenda

- Context and motivations
- **MBT for Conformance Testing – GlobalPlatform compliance program**
  - Introducing GlobalPlatform
  - Model-Based Compliance Test Generation: Process and results
- Application to SCA 2.2.2
- Lessons learned and Conclusion

GlobalPlatform Positioning

GlobalPlatform is *the* standard for managing applications on secure chip technology



Across several market sectors and in converging sectors

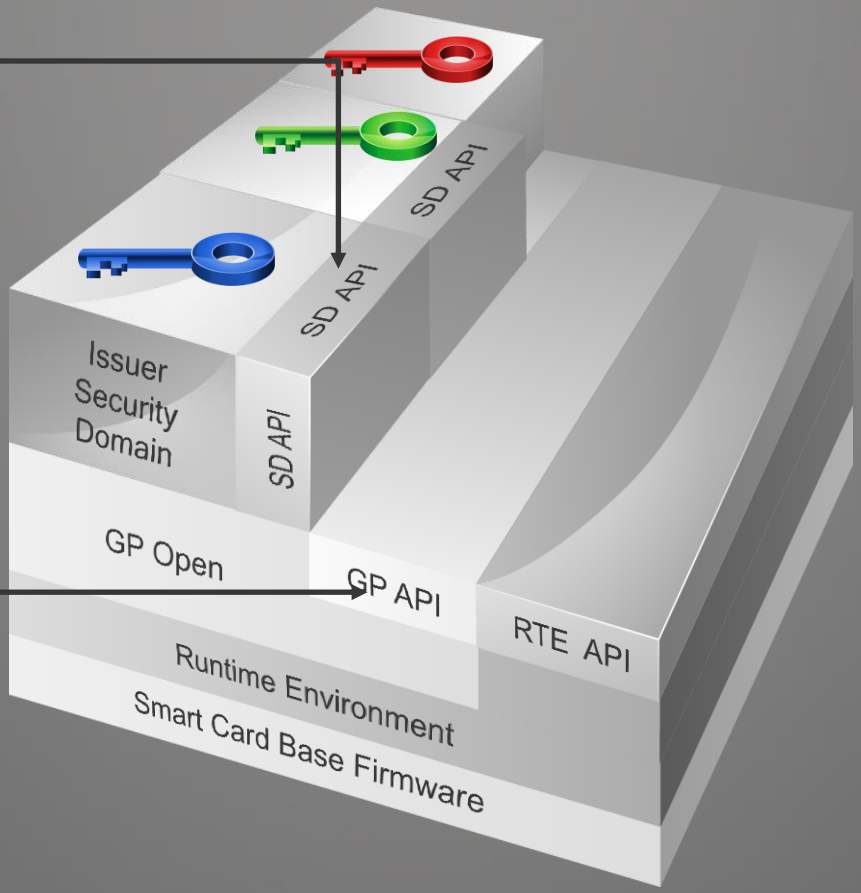
Introducing GlobalPlatform standards...

- Since 1999, With GlobalPlatform standards:
- Create once based on -
  - Stable and interoperable Application Programming Interfaces (APIs)
  - Stable security requirement
- Deploy ‘everywhere’

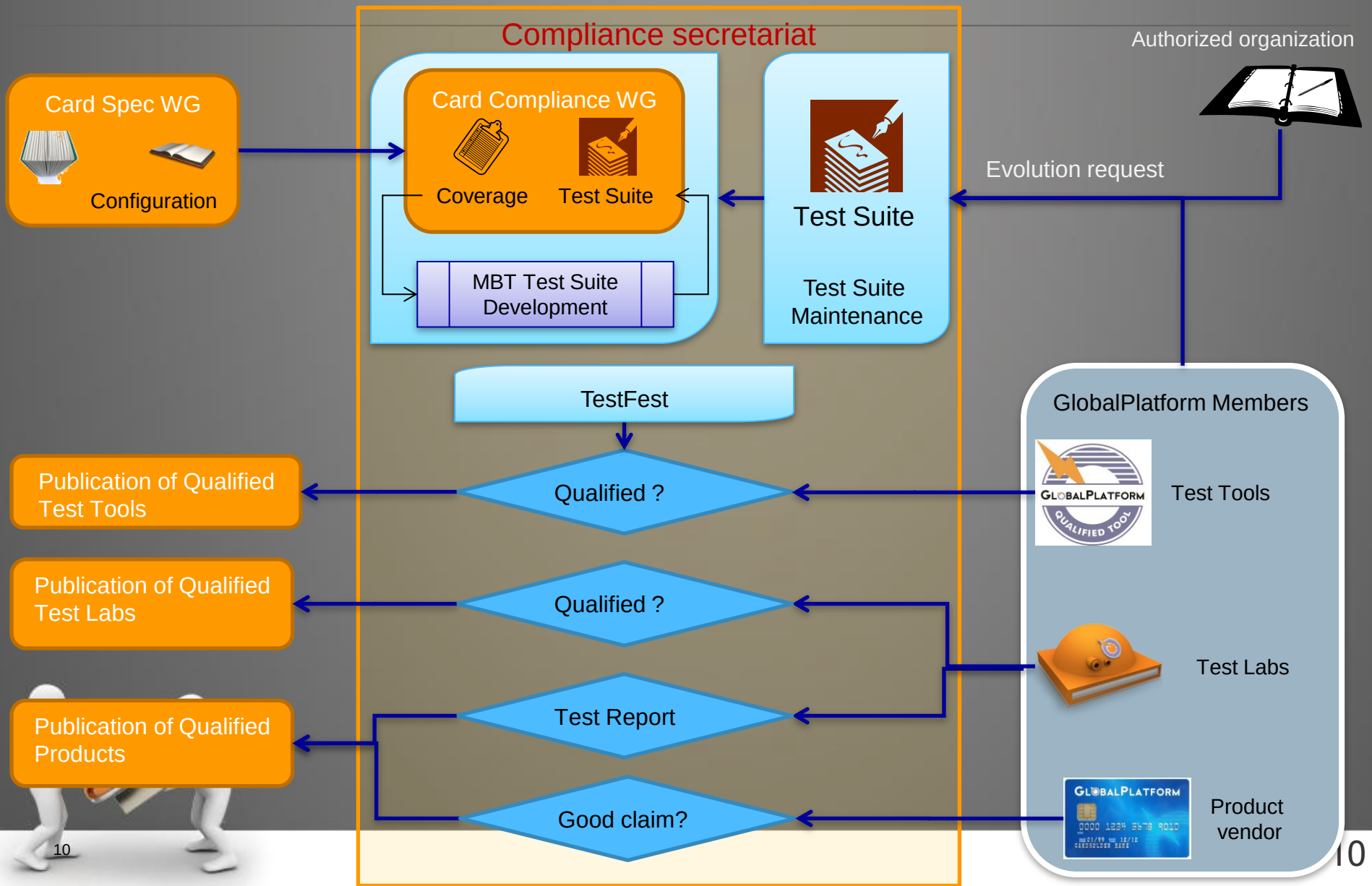
GlobalPlatform Card Framework Reminder

GlobalPlatform Specification

Commands APDU (ISO 7816)



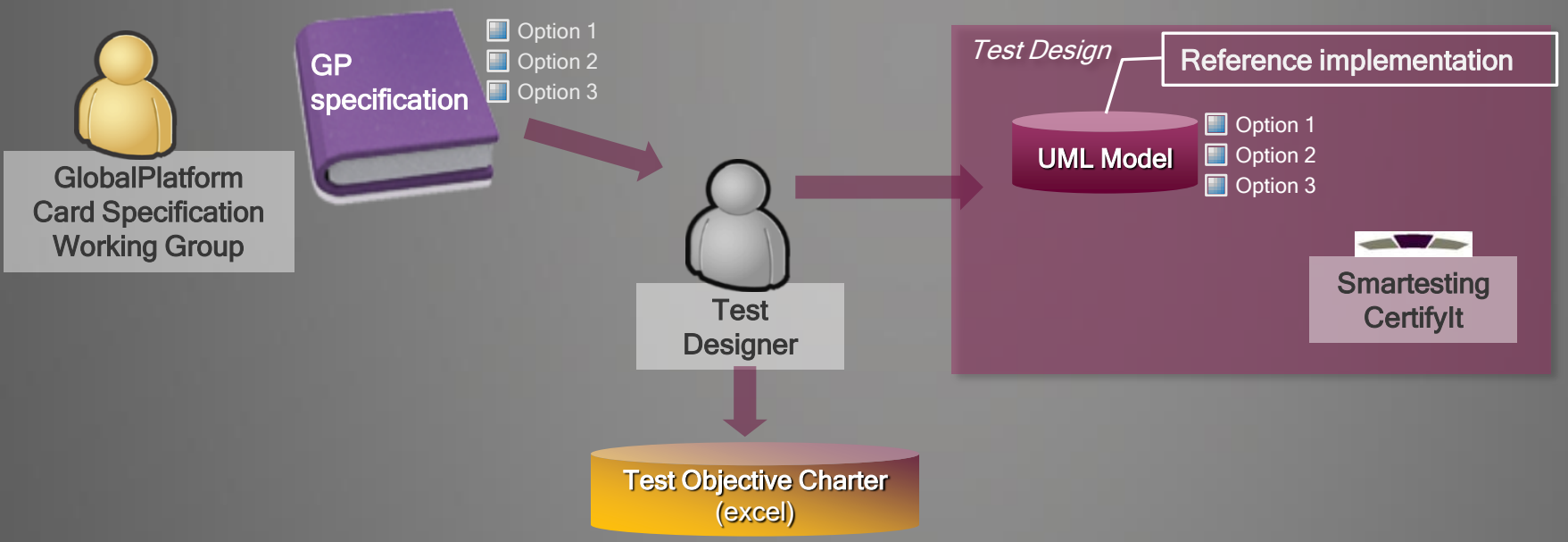
GlobalPlatform Compliance Program Process



Challenges about Test Suite Development

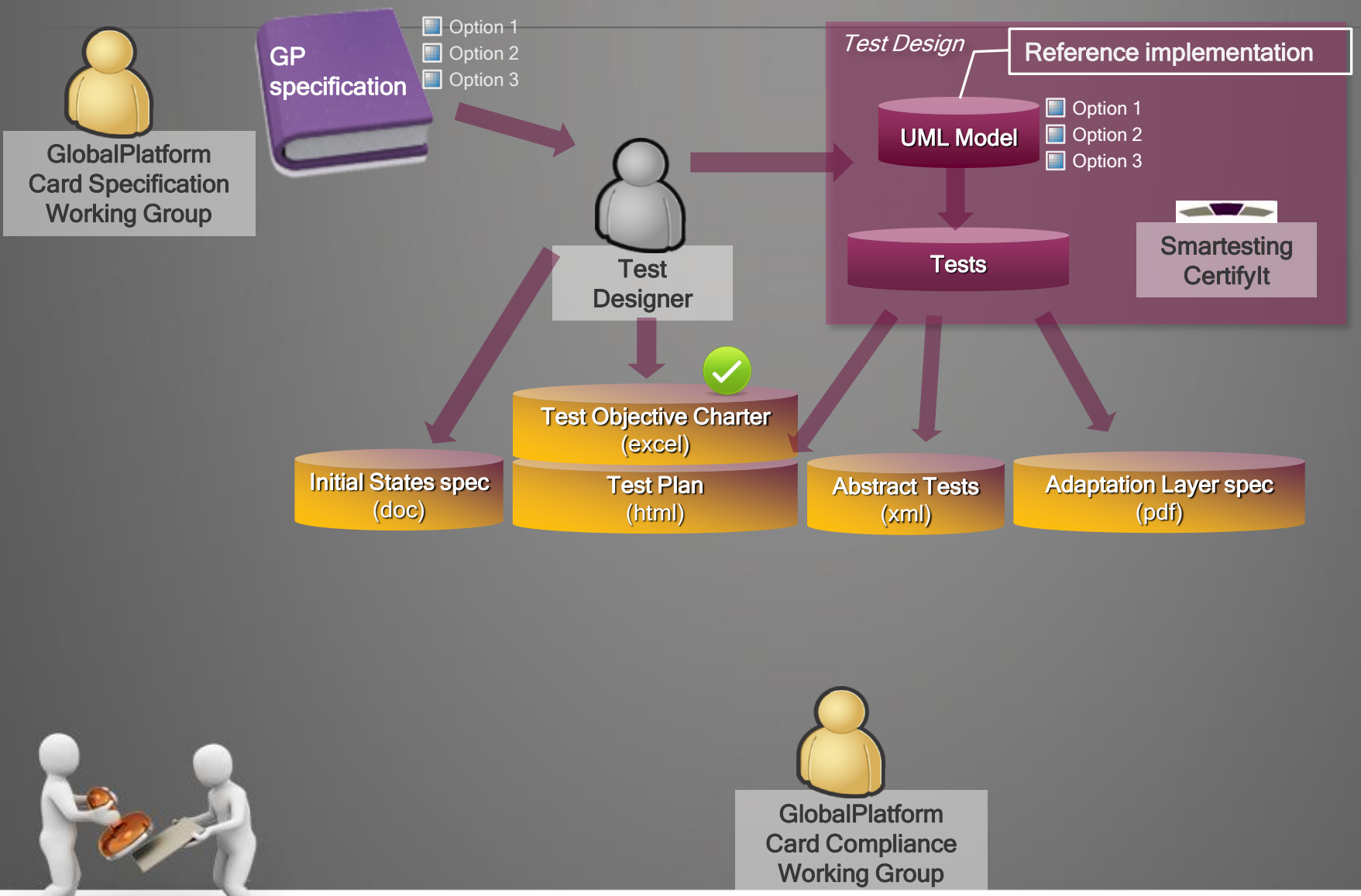
- The test suite shall be open to any test execution tools
- The variants in specification and in test suite shall be managed efficiently
  - Generic specification with various specific configurations spec
  - Test Suites are developed for each specific configurations spec
  - Challenge is to reuse the generic part and to manage the specific part in an efficient manner

From Specifications to Test Plan deliverable

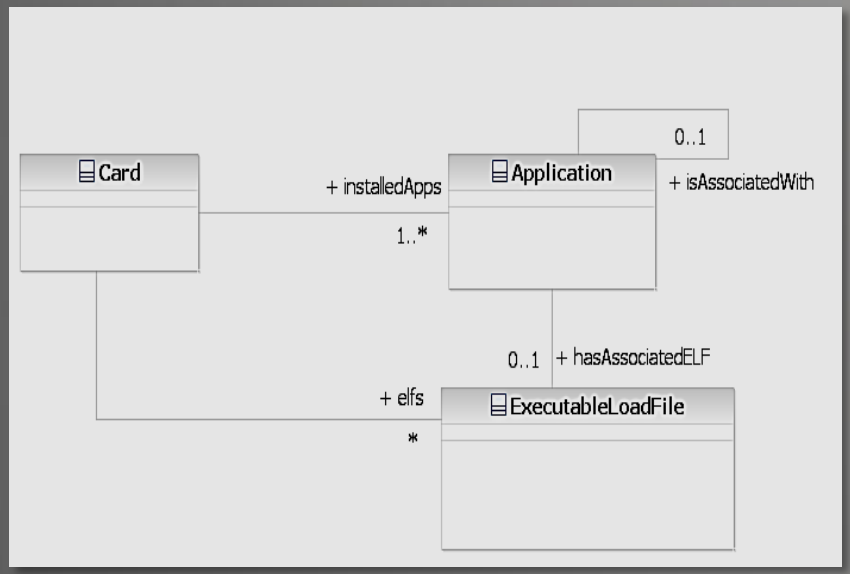


GlobalPlatform
Card Compliance
Working Group

From Specifications to Test Plan deliverable

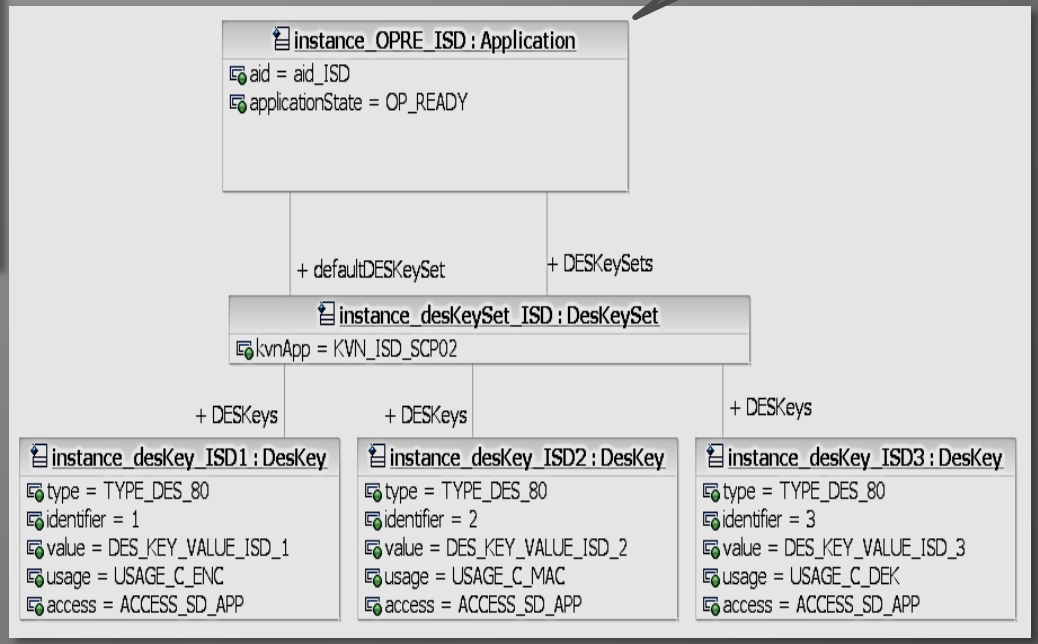


The UML model



The entities

The initial state



The Test Objective Charter

Test Objective Charter (XLS)

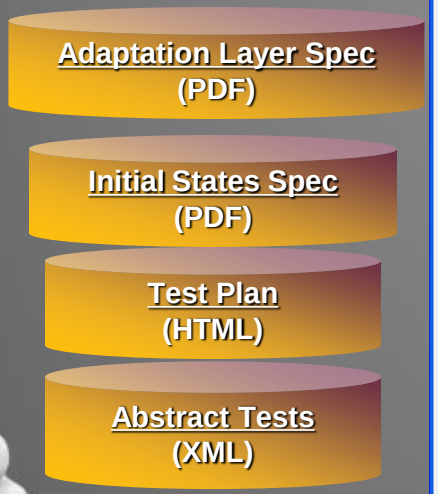
Requirement extracted from Specification

Specific cases 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				

Deliverables

- One deliverable in HTML containing
  - All test compliance packages



Test Plan (HTML) (3/3)

- A test detailed

Test sequence detailed

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

Technical results (1/2)

- 15 active Compliances Test Suites
- 6 000+ tests in Compliance Test Suites
- 5 Test Tools are qualified
  - Each Test Tools is importing the generated Test Plan as XML

Technical results (2/2)

- Various domains of application
  - GlobalPlatform (Card)
    - Configurations derived from a generic spec GP 2.2
      - Telecom configuration : UICC + UICC Contactless extension
      - Banking configuration : Mapping Guidelines / Basic Financial
      - Identity configuration : ID
      - Embedded Secure Element configuration : SE
      - Generic configuration : Common Implementation
    - SWP/HCI Test Suite from **ETSI** specifications
  - GlobalPlatform (Device)
    - Trusted Execution Environment : Client API + Internal API
    - Open Mobile API Specifications from **SIMalliance consortium**
  - GlobalPlatform (System)
    - Trusted Service Manager Compliance : multi-component testing

Lessons Learned from GP Experience

- Test case generation is an automated process and so **more predictive and less error-prone** than manual processes.
- It gives to the generated test cases a **clear functional coverage** metrics from the viewpoint of the TOC.
- All generated assets (XML and HTML test suites, Adaptation Layer Specification) are **kept in sync** because 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.

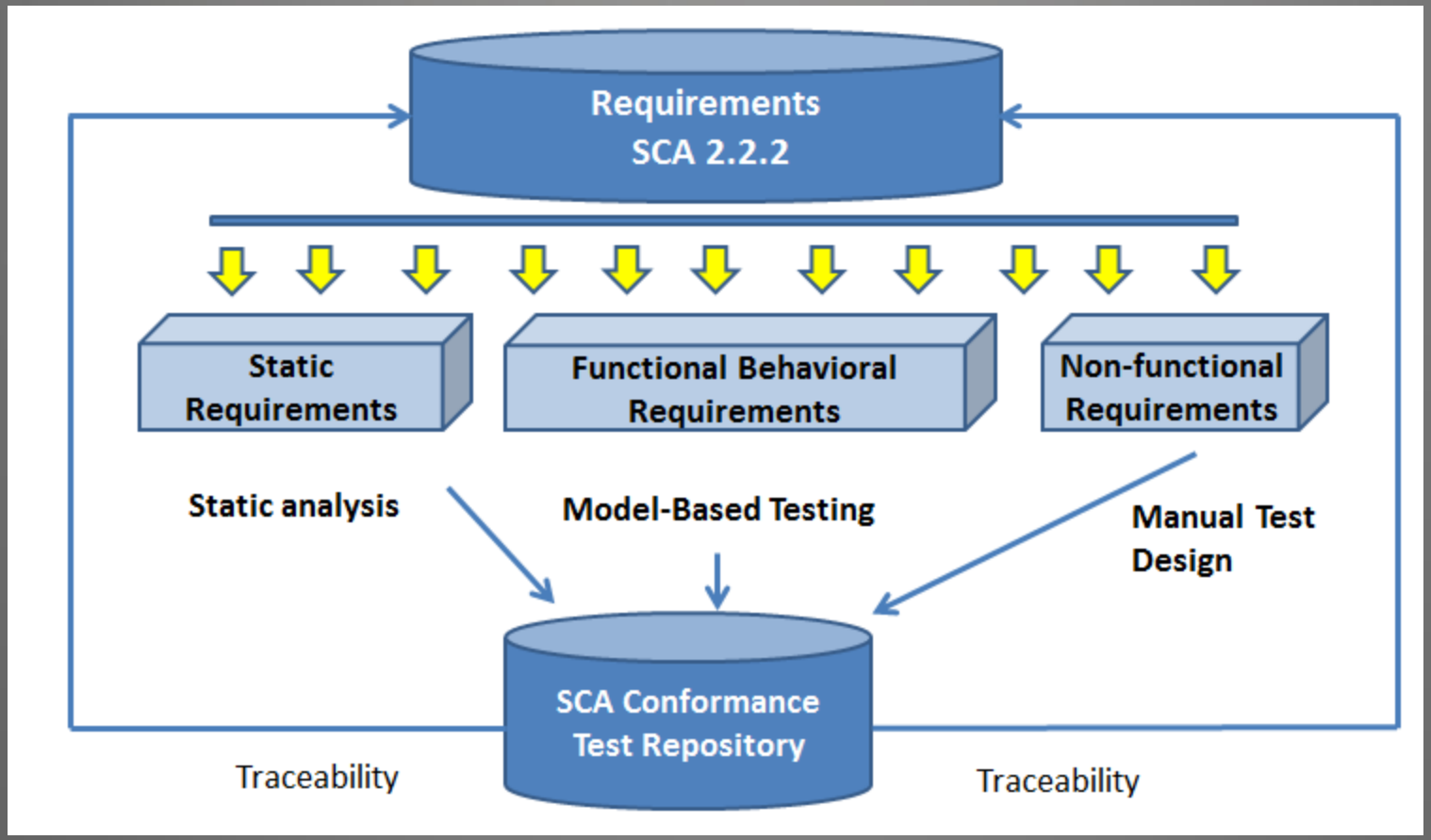
Summary on GlobalPlatform

- From 2007 till present, the GP Compliance Program has been using Model-Based Testing to produce its compliance test suite, which is now providing a unique value:
  - the test suite (means 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, 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, currently for system compliance

Agenda

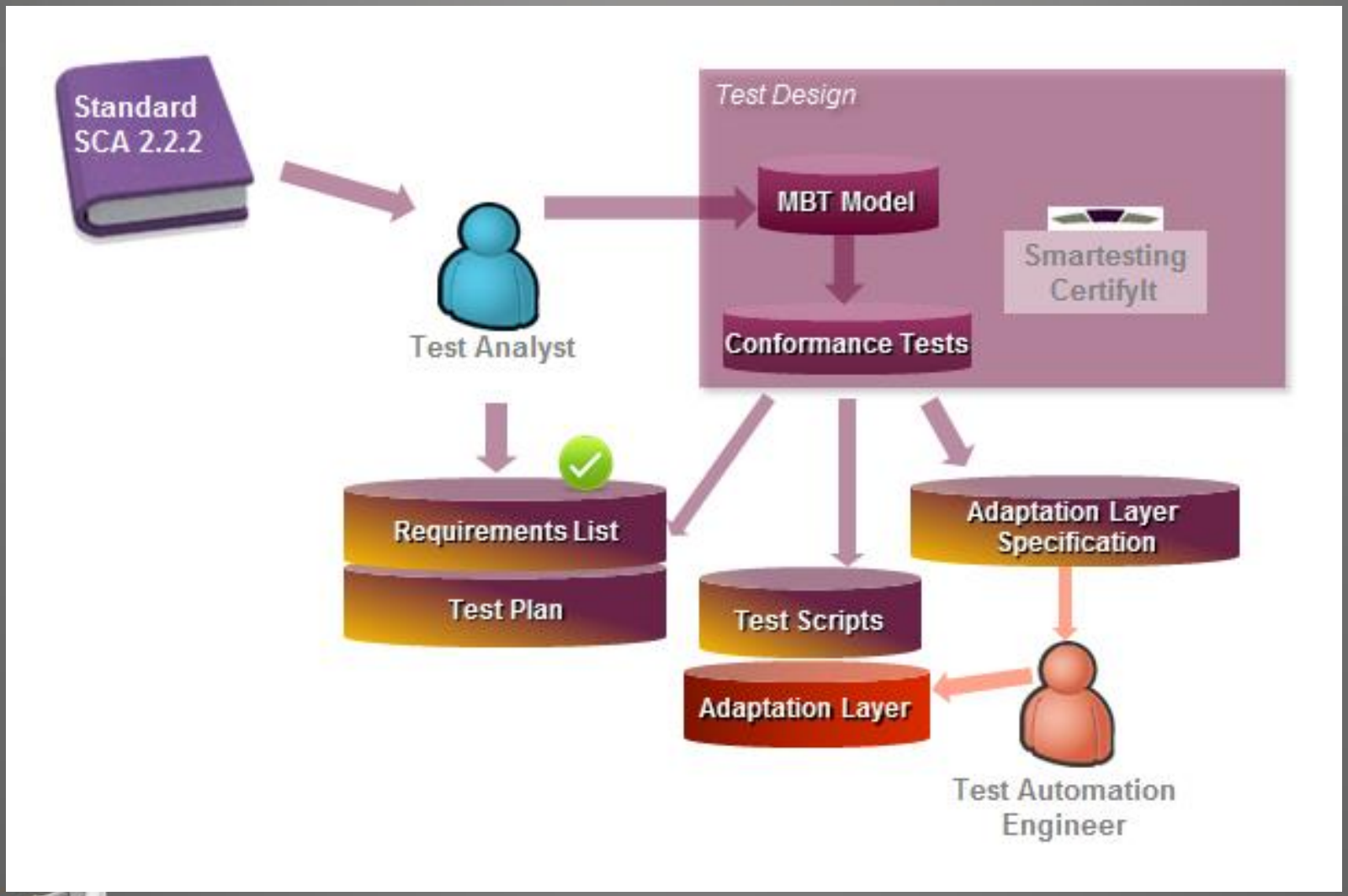
- Context and motivations
- MBT for Conformance Testing – GlobalPlatform compliance program
- **Application to SCA 2.2.2**
- Lessons learned and Conclusion

SCA conformance testing - MBT postionning



Both for Platforms and Waveforms

SCA Conformance Testing – MBT Process



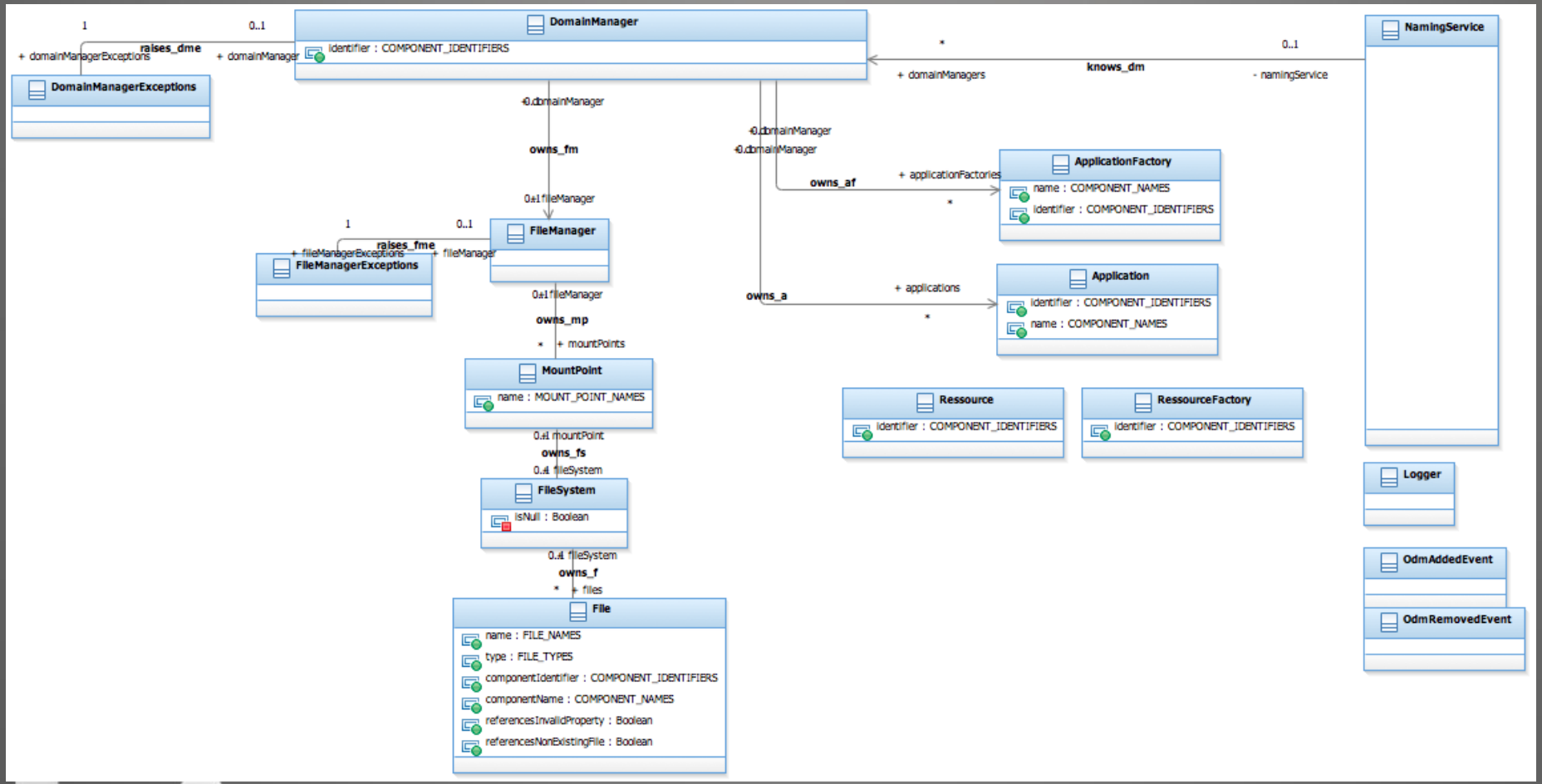
Proof of Concept - SCA 2.2.2 scope

Requirement Tag	Criterion Tag	Requirement/Criterion Text	Section Number	Test Method	JTAP Test Case Name	Manual Test Case Number
SCA 2.2.2 Specification - Main Body						
AP0011		A log producer shall only output log records that contain an enabled CosLwLog::LogLevel value.	3.1.2.2.1	Manual		APP_TC_001
AP0012		Log producers shall use their component identifier attribute in the producerId field of the CosLwLog::ProducerLogRecord.	3.1.2.2.1	Manual		APP_TC_001
AP0013		Log producers and CF components that are required by this specification to write log records shall operate normally in the absence of a log service or in the case where the connections to a log are nil or an invalid reference.	3.1.2.2.1	Manual		APP_TC_001
AP0063		A component (e.g., Resource, DomainManager, etc.) that consumes events shall implement the CosEventComm PushConsumer interface.	3.1.2.3.1	Manual		APP_TC_029
AP0064		A component (e.g., Resource, Device, DomainManager, etc.) that produces events shall implement the CosEventComm PushSupplier interface and use the CosEventComm PushConsumer interface for generating the events.	3.1.2.3.1	Manual		APP_TC_029
AP0065		A producer component shall not forward or raise any exceptions when the connection to a CosEventComm PushConsumer is a nil or invalid reference.	3.1.2.3.1	Manual		APP_TC_029
AP0069		The connectPort operation shall make a connection to the component identified by its input parameters.	3.1.3.1.1.5.1.3	Automated	ConnectPort	APP_TC_015
AP0069	C002	A port may support several connections. The input connectionId is a unique identifier to be used by the disconnectPort operation when breaking a specific connection.	3.1.3.1.1.5.1.3	Manual		APP_TC_015
AP0070[2]		The connectPort operation shall raise the InvalidPort exception when the input connection parameter is an invalid connection for this port.	3.1.3.1.1.5.1.5	Automated	ConnectPort InvalidPort Exception	APP_TC_014
AP0070	C004[2]	The InvalidPort exception indicates one of the following errors has occurred in the specification of a Port association: 1. errorCode 1 means the Port component is invalid (unable to narrow object reference) or illegal object reference.	3.1.3.1.1.3.1	Automated	ConnectPort InvalidPort Exception	APP_TC_014
AP0071		The connectPort operation shall raise the OccupiedPort exception when unable to accept the connections because the port is already fully occupied.	3.1.3.1.1.5.1.5	Automated	ConnectPort Occupied Port Exception	APP_TC_015
AP0072		The disconnectPort operation shall break the connection to the component identified by the input connectionId parameter.	3.1.3.1.1.5.2.3	Automated	DisconnectPort Test	APP_TC_012
AP0073[2]		The disconnectPort operation shall raise the InvalidPort exception when the input connectionId parameter is not a known connection to the Port component.	3.1.3.1.1.5.2.5	Automated	DisconnectPort Test	APP_TC_012

Figure 5: Studied Excerpt of SCA 2.2.2 Application Requirements List Version 2.2.

Test Objective Charter

MBT Model- Class diagram

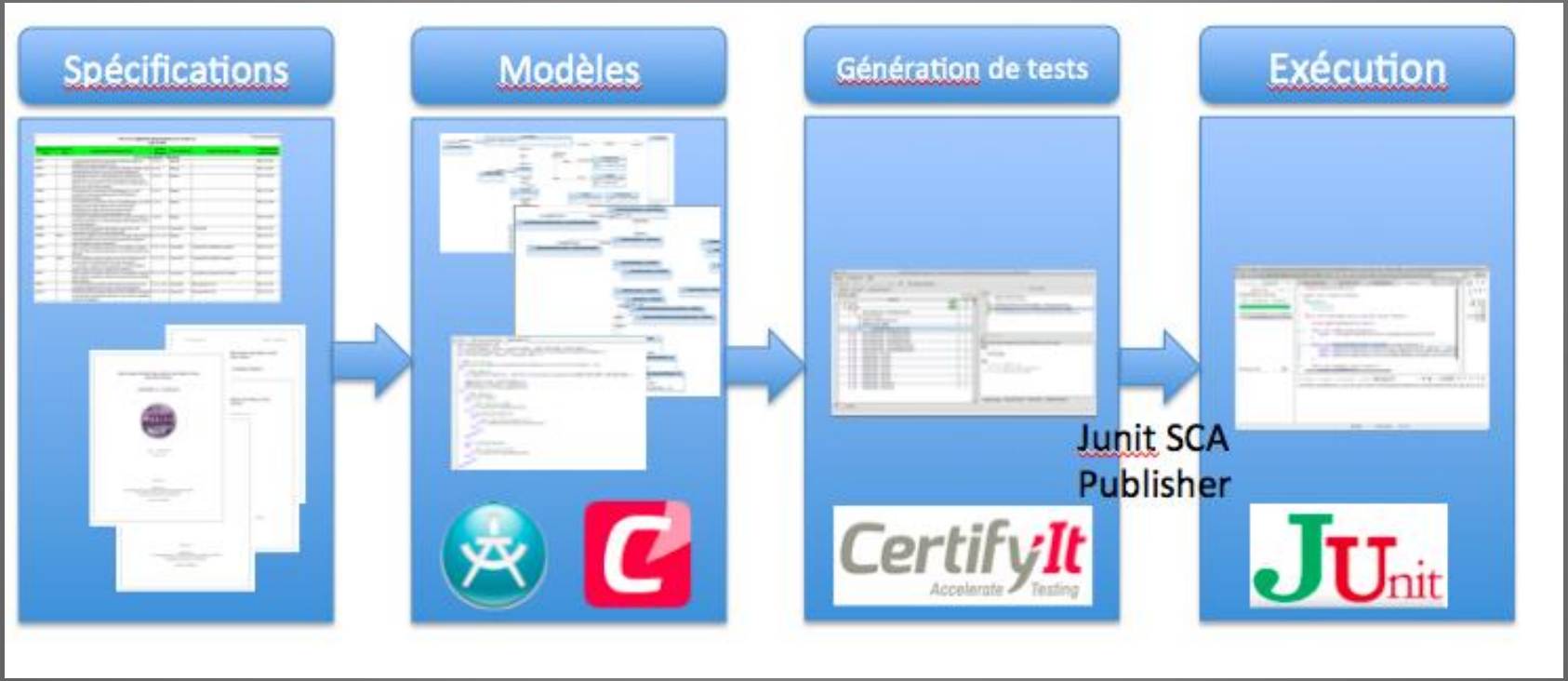


MBT Model - OCL Constraints of the Operation *mount()*

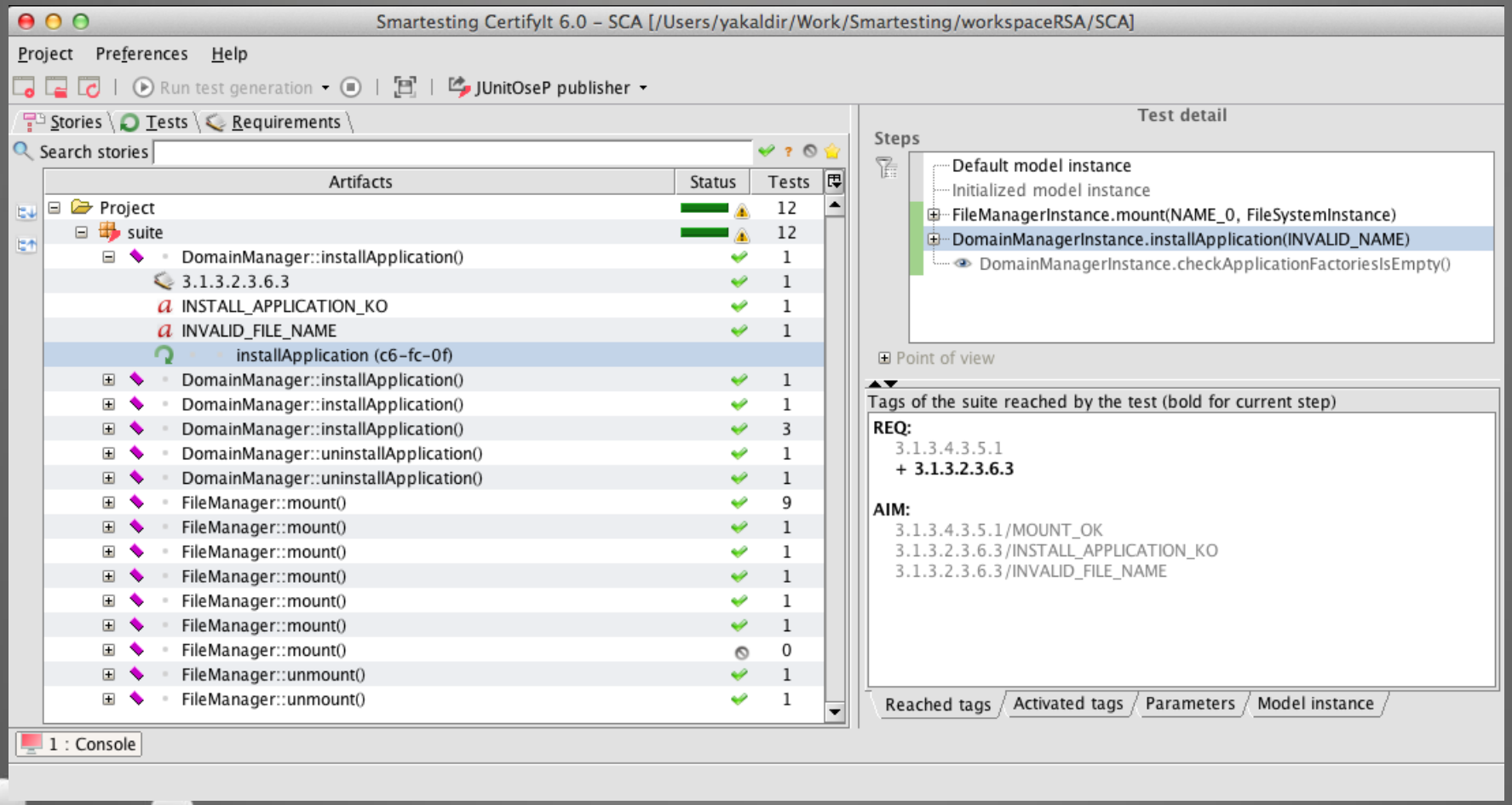
*Main *DiagrammeObjet1 mount

```
self.resetExceptions() and
let invalidFileName : Boolean = (mountPointName = MOUNT_POINT_NAMES::INVALID_NAME) in
let mountPointAlreadyExists : Boolean = (self.mountPoints->exists(mp|mp.name=mountPointName)) in
let invalidFileSystem : Boolean = fileSystem.isNull in

---@REQ: 3.1.3.4.3.5.1
if (not(invalidFileName) and not(mountPointAlreadyExists) and not(invalidFileSystem)) = true
then
    ---@AIM: MOUNT_OK
    let newMountPoint:MountPoint = MountPoint.allInstances()->any(mp|mp.name=MOUNT_POINT_NAMES::UNDEFINED_NAME) in

    newMountPoint.name = mountPointName and
    newMountPoint.fileSystem = fileSystem and
    self.mountPoints->includes(newMountPoint)
else
    ---@AIM: MOUNT_KO
    if (invalidFileName)
    then
        ---@AIM: INVALID_FILE_NAME
        self.raiseInvalidFileNameException()
    else
        if (mountPointAlreadyExists)
        then
            ---@AIM: MOUNT_POINT_ALREADY_EXISTS
            self.raiseMountPointAlreadyExistsException()
        else
            true
        endif
    endif

    and if (invalidFileSystem)
    then
        ---@AIM: INVALID_FILE_SYSTEM
        self.raiseInvalidFileSystemException()
    else
        true
    endif
endif
```

MBT for SCA conformance testing - Tool chain



Generated tests



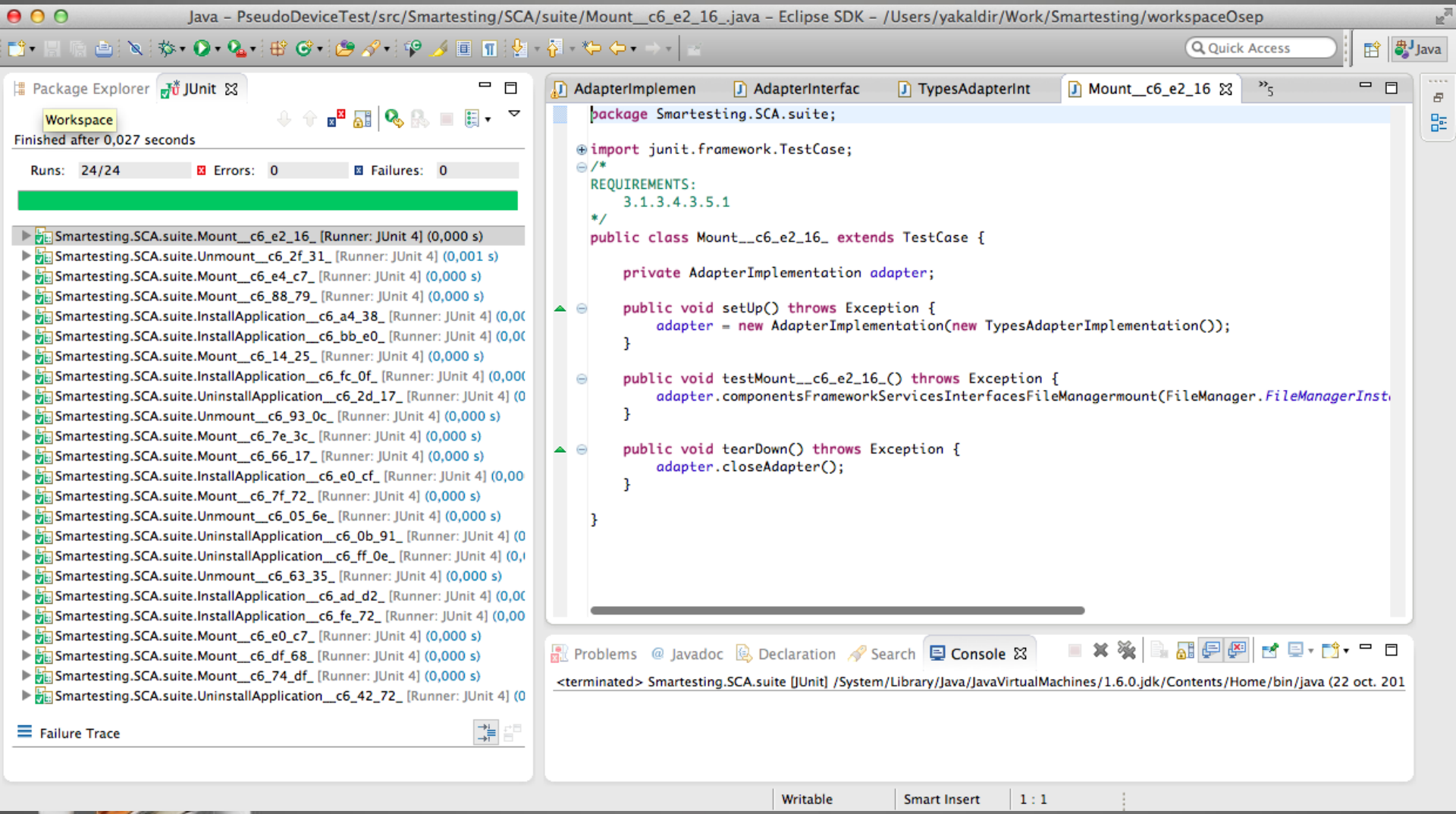
Generated Scripts

```
package Smartesting.SCA.suite;

import junit.framework.TestCase;
/*
REQUIREMENTS:
    3.1.3.2.3.6.3
    3.1.3.4.3.5.1
*/
public class InstallApplication__c6_a4_38_ extends TestCase {

    private AdapterImplementation adapter;

    public void setUp() throws Exception {
        adapter = new AdapterImplementation(new TypesAdapterImplementation());
    }

    public void testInstallApplication__c6_a4_38_() throws Exception {
        adapter.componentsFrameworkServicesInterfacesFileManagermount(FileManager.FileManagerInstance, MOUNT_POINT_NAMES.NAME_0, FileSystem.FileSystemInstance);
        adapter.componentsFrameworkControlInterfacesDomainManagerinstallApplication(DomainManager.DomainManagerInstance, FILE_NAMES.INVALID_NAME);
        adapter.componentsFrameworkControlInterfacesDomainManagercheckApplicationFactoriesIsEmpty(DomainManager.DomainManagerInstance);
    }

    public void tearDown() throws Exception {
        adapter.closeAdapter();
    }
}
```

Adapatation Layer for atomated test execution

```
import SCA.TypesDeclaration.COMPONENT_NAMES;
import SCA.TypesDeclaration.FILE_NAMES;
import SCA.TypesDeclaration.MOUNT_POINT_NAMES;
import SCA.TypesDeclaration.OdmAddedEvent;
import SCA.TypesDeclaration.OdmRemovedEvent;
import SCA.TypesDeclaration.SOURCE_CATEGORY_TYPE;
import SCA.TypesDefinition.COMPONENT_IDENTIFIERS;
import SCA.TypesDefinition.FileSystem;

public class AdapterImplementation implements AdapterInterface {

    static String tab = "    ";
    PrintStream ps;
    int ok = 0;
    int ko = 0;
    private String fileLogs = "./fileLog.txt";
    private FileOutputStream fileLog;
    private PrintStream pfileLog;
    private int appFactorySeqLength = 0;

    @Override
    public void componentsFrameworkControlInterfacesDomainManagercheckApplicationFactories(
            SCA.TypesDefinition.DomainManager receiverInstance,
            SCA.TypesDefinition.ApplicationFactory applicationFactory,
            SCA.TypesDeclaration.COMPONENT_IDENTIFIERS out_identifier,
            COMPONENT_NAMES out_name) throws Exception {
        // TODO Auto-generated method stub

    }
    @Override
    public void componentsFrameworkServicesInterfacesFileManagerunmount(
            SCA.TypesDefinition.FileManager receiverInstance,
            MOUNT_POINT_NAMES mountPoint) throws Exception {
        final FileManager fm = TypesAdapter.getConcreteValue(receiverInstance);
        try {
            fm.unmount(mountPoint.toString());
            ps.println("OK : File System unmounted");
            ok++;
            listMountedFileSystems(fm);
        } catch (NonExistentMount e) {
            ps.println("KO : Non Existent Mount Exception : File System");
            ko++;
        }
    }

    private void listMountedFileSystems(FileManagerOperations fm) {
        ps.println(tab + "List of Mounted Types : " + fm.getMounts().toString());
        for (int i = 0; i < fm.getMounts().length; i++) {
            ps.println(tab + "Mount Point : " + fm.getMounts()[i].mountPoint
                    + " \n File System : " + fm.getMounts()[i].fs.toString());
        }
    }
```

Test Execution using JUnit



Lessons learned and Conclusion

- Model-Based Testing is adequate for SCA conformance testing
- This process targets functional behavioral SCA requirements
- The modeling notation used (UML / OCL) was suitable to model targeted SCA requirements
- The MBT tool-chain helps to automate test generation (documented test repository, traceability and executable test scripts)
    - Increased productivity for the development of the conformance test suite
    - Better control about what is really tested and how in the conformance test suite

Thank you for attention !
Any question ?

Eddie.Jaffuel@eConsult.fr

Bruno.Legeard@femto-st.fr
Bruno.Legeard@smartesting.com