Application of TTCN-3 Test Language to Testing Information Systems in eHealth Domain*

Diana Elena Vega[1], George Din[2], Ina Schieferdecker[1,2]
[1]*Technical University of Berlin, ETS, Germany*
[2]*Fraunhofer FOKUS, Germany*
vegadiana@cs.tu-berlin.de, george.din@fokus.fraunhofer.de, ina@cs.tu-berlin.de

Abstract

The adoption of standards in eHealth domain, such as Health Level 7 (HL7) used for data representation, and Integrating Healthcare Enterprise (IHE) [4] for describing interactions between actors, is an important step in enabling healthcare systems to interoperate with each other. This paper addresses the challenges of interoperability testing of different HL7 based systems and presents a test framework based on TTCN-3 language.

1 Introduction

Healthcare Information Systems (HIS) are based on standards for structuring patient data and exchanging the data between different healthcare facilities (e.g., radiology, laboratory). We focus in this paper on information systems built on Health Level 7 (HL7) [3] and compliant with Integrated Healthcare Enterprise (IHE). HL7 standard defines a common message structuring scheme for all messages used in medical information systems. IHE defines the use cases which HIS implementers should follow. These standards are the basis for the interoperability testing.

So far, only little attention has been given to a common methodology for testing HIS. The development of our approach is based on the standardised Testing and Test Control Notation version 3 (TTCN-3) [1] test technology and it is the first attempt to employ this test technology to testing HIS systems.

We design a test system capable of simulating the interacting parties when one or a set of particular components need to be tested. The challenge of this approach is to automatically configure the test platform to simulate an interoperability scenario by instantiating test components programmed in advance to simulate the behaviour of particular actors. In our approach we test the system under test (SUT) against a reference test system (TS) in order to check if it satisfies the interoperability requirements. It has though the big advantage that it can be used in the lab with no need for real components to interact with.

The rest of the paper is structured as follows. After related work section we introduce the Patient Care Device (PCD) IHE profile case study in Section 3. Section 4 presents the TTCN-3 based test framework. The last section finalises the paper with our conclusions.

2 Related Work

Test solutions for interoperability are currently under research or development. The solution provided in [10] is designed to verify the input and output of Electronic Health Record (EHR) data against the standards and criteria identified by the Certification Commission for Healthcare Information Technology (CCHIT). A similar approach is proposed in [11]. However, we consider these approaches difficult to apply in the early stages of testing as they require complex setups and, especially, the presence of all interacting parties. Compared with these approaches, in our methodology we propose a different technique in which the test systems simulate the components that the SUT needs to interact with. This way, the interoperability is always tested against a reference implementation.

3 IHE PCD Case Study

The design concepts and the implementation of our test system (TS) have been applied to the Patient Care Device (PCD) [5] IHE profile case study. PCD profile standardises the communication scenarios and flows

*This work was partially supported by the TestNGMed Project[6]

between medical devices directly connected to a monitored care unit (e.g., blood pressure sensors), and other units from the medical environment interested in receiving data from those medical devices.

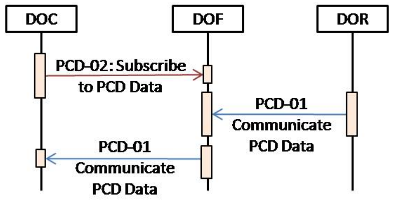
DOC
DOF
DOR
PCD-02: Subscribe to PCD Data
PCD-01 Communicate PCD Data
PCD-01 Communicate PCD Data

Figure 1. IHE PCD Profile

Figure 1 presents in a sequence diagram the PCD actors and the sequence of possible interactions between them. The Device Observation Reporter (DOR) has the role of monitoring all medical devices from which data needs to be included into the system (ventilator, blood pressure sensor, etc.), and the role of converting that data into digital format, if necessary, acting as a data provider for the other actors of the PCD profile. The Device Observation Filter (DOF) is the entity responsible for handling a connection between a Device Observation Consumer (DOC) of the medical data (laboratory, medical clinic, etc.) and the DOR, in charge of providing that data. The DOF manages subscriptions and offers to the DOC subscribers the possibility to receive a subset of the data stream, according to their needs and subscription predicates. The DOC is the entity for which the system has been designed, and in the real world it can be the medical clinic, the laboratory or any entity that requests data about patients.

An instance of this diagram may involve an arbitrary number of actors. Multiple DORs can be involved at the same time, while many DOC consumers may receive data in parallel. The DOF is responsible for handling all subscriptions and ensures that the DOCs receive the required data.

The actors communicate through two types of transactions. PCD-02 is used by a DOC to subscribe/unsubscribe to a DOF for PCD data. The DOF receives the subscribe requests, and set the adequate filters. PCD-01 is used to communicate PCD data from a DOR to a DOC as filtered by DOF.

4 The TTCN-3 Test Framework

The TTCN-3 [1] standardised testing language is a textual language, which is for testers easy to learn and to use. It has the look and feel of a programming language but has been designed for testing including artifacts required for testing.

In order to create the TTCN-3 based test system we first created a mapping of HL7 PCD message types to TTCN-3 *data types*. Next, we defined algorithms to derive *test configurations* and *test behaviours* out the PCD interaction scenarios. In order to get an executable test system based on TTCN-3, an *adaptation component* [2] called *Adapter* and an *encoding/decoding component* called *CoDec* have been provided. These components are used by the execution environment (we used TTworkbench [7]), in order to adapt the data and the requests to the SUT and its interfaces.

For the selected PCD case study, the SUT consists of DOF and DOR actors, while the test system emulates the DOC actor. The SUT is the Polybench tool [8], a medical software for electro-physiology, enhanced during the project with the DOF and DOR capabilities. For writing, compiling and executing the TTCN-3 test scripts, TTworkbench [7] IDE was used.

HL7 to TTCN-3 Data Type Mapping. The mapping rules from HL7 types to TTCN-3 data types follow closely the structure of the HL7 message format (we described these mapping rules in [12]). A HL7 message has a tree-like structure and it maps to a tree-like structure represented in the TTCN-3 syntax by means of *record*, *record of*, etc., types. The TTCN-3 also supports attributes and constraints such as *optionality* of leaves, *length*, *cardinality* of segments, etc.

Listing 1. PCD-02 Type in TTCN-3

```
type record PCD_02_Message_Type {
  MSH_Segment_Type MSH_Segment,
  QPD_Segment_Type QPD_Segment,
  RCP_Segment_Type RCP_Segment,
  DSC_Segment_Type DSC_Segment optional };
type record QPD_Segment_Type {
  CE_HL70471   Message_Query_Name,
   ... };
```

An example of PCD-02 data representation in TTCN-3 language is represented in Listing 1. In contrast to the mandatory segments, the optional segments are marked with the keyword *optional*. We mapped the *cardinality* concept from HL7 by using the array type (*record of*) whose length is restricted by the cardinality order given by the PCD profile.

To keep a clear test suite and module structure of the huge TTCN-3 specification, we developed and applied a set of guidelines for writing and structuring the TTCN-3 specifications.

Generation of Configuration Parameters. Multiple interaction scenarios may correspond to the same IHE profile. Consequently, an IHE profile specification translates into many test scenarios. The test configurations (ports, components and their parameters) and the test scripts have been generated out of test scenarios designed in UML Sequence Diagrams (UML-SD) which are derived from UML Activity Diagrams (UML-AD) corresponding to an IHE profile. The technical realisation of the generation algorithms has been achieved within the .getmore tool [9].

Listing 2. TTCN-3 Module Parameters

```
modulepar {
  charstring add_SUT_Filter:="127.0.0.1:10000";
  charstring portPTC_DOC_PRESSURE:="15001";
  charstring portPTC_DOC_VOLUME:="15002";
}
```

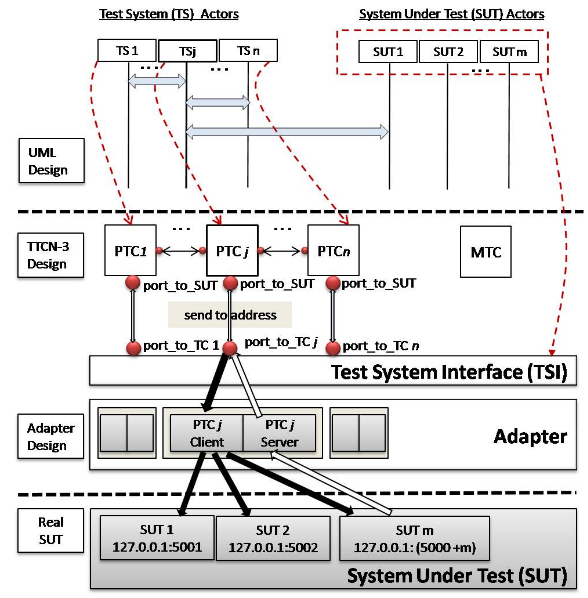

Figure 2. Test Scenario mapping to TTCN-3 Test Configurations

An IHE profile implementation encapsulates configuration parameters such as the IP address and the port where the PCD actors are running on. Therefore, the test scenarios need to be annotated with such information. In TTCN-3 such parameters are represented as *module parameters*. Listing 2 shows some module parameters required for testing the DOF actor. They basically contain IP and ports necessary to locate the computers where actors are running.

Generation of Test Components. The mapping of actors to test components is illustrated in Figure 2. A TTCN-3 testcase runs on top of a *main test component* (MTC) that can create other *paralel test components* (PTCs). The *test system interface* (TSI) is a special type of component and it represents an abstraction of the SUT at TTCN-3 specification level. We distinguish between tested actors (SUT1, SUT2, . . . , SUTm) and actors simulated by test system (TS1, . . . TSj, . . . TSn). Each TS actor in the test scenario translates into a TTCN-3 test component type. The instances are called PTCs. The SUT actors will be mapped into only one TTCN-3 component type whose instance is used as TSI. Additionally, an empty-body MTC component type is generated. This component is used to start the test case and create the other PTCs. An example of a generated test configuration is shown in Listing 3.

Listing 3. TTCN-3 Test Configuration - Component Types

```
type component DOC_Comp_Type {
  port HL7Port_Type p_HL7_SuT;}
type component TSI {
  port HL7Port_Type p_HL7_DOC;}
type component MTC { }
```

Generation of Communication Ports. We differentiate between the situations where the component type containing the port definition will be used as a) PTC or as b) TSI component.

a) We associated one TTCN-3 component type to each actor defined in the test scenario. For clarification, we analyse the *PTCj* component in Figure 2. This component type contains one port instance for each non SUT actor (PTC1, . . . , PTCn). These ports are used for the communication between the actors. For the interaction with SUT actors (SUT1, SUT2, . . . , SUTm), only one port instance *port_to_SUT* is defined on the PTCj component. In this view, the SUT is seen as only one unit composed by SUT actors. The different SUT actors are addressed by using the *send to address* construct provided by TTCN-3, where address is substituted by the value of the TTCN-3 module parameter corresponding to the appropriate SUT entity (IP:port).

b) The TSI component type defines one port instance for each non SUT actor (mapped as PTC) interacting with at least one SUT actor. In Figure 2, the TSI component has one port to each PTC called *port_to_TC*.

Generation of TTCN-3 Behaviour Functions. The behaviour of one actor is determined by the sequence of interactions it has with the other actors. The interaction is mapped to a TTCN-3 test behaviour. On the test system side, this involves parallel behaviours of many (SUT and TS) actors. In Figure 2, the MTC component instantiates each PTC component and starts it with a test behaviour describing how that PTC acts as the actor it simulates.

Listing 4 shows an example of a TTCN-3 behaviour function corresponding to a DOC actor. It sends a subscribe message (line 4) to the DOF with a request for patient data. At line 6 a timer is started. The *alt* construct starting at line 7 is used to handle the possible reactions of the SUT. Line 8 describes the alternative when the subscription succeeded and the PCD received one PCD-01 message. The second alternative, at line 11, describes the situation when the SUT sends wrong data (unexpected), e.g., the receiving ID does not correspond to the requested subscription. The last alternative reflects the situation when the SUT does not forward any data from the reporter. The wait duration is validated by the *replyTimer*. The second and third alternatives stop the test behaviour (i.e., *mtc.stop*) and set the verdict to *fail*. The first alternative jumps to line 17 for unsubscription request. The unsubscription template is defined by setting the *Action_Code* to value "D" (line 18).

Listing 4. TTCN-3 Behaviour Example

```
1  function Consumer_PRESSURE_Behaviour()
2  runs on Consumer_PRESSURE_Comp_Type {
3   timer replyTimer;
4   pt_data_HL7_SuT.send(PCD_02_REG_PRESSURE)
5   to address_Filter;
6   replyTimer.start(20.0);
7   alt {
8    [] pt_data_HL7_SuT.
9     receive(PCD_01_Receive_Template_DOF){
10    replyTimer.stop;}
11   [] pt_data_HL7_SuT.receive{
12    replyTimer.stop; setverdict(fail);
13    mtc.stop; }
14   [] replyTimer.timeout{
15    setverdict (fail); mtc.stop;}
16  }
17  PCD_02_REG_PRESSURE.
18    QPD_Segment.Action_Code:="D";
19  pt_data_HL7_SuT.send(PCD_02_REG_PRESSURE)
20    to address_Filter;
21 }
```

Implementation of the Adapter and CoDec. The role of the adapter is to handle PCD transactions between TS simulated actors and the SUT actors. For each actor simulated by the test system, the adapter defines a pair of *server* and *client* components (see Figure 2). The *client component* is used to dispach messages to the desired SUT actor address. The *server component* is responsible for handling connections from SUT actors. This design allows an arbitrary number of PTCs without changes needed in the adapter. The role of the CoDec is to translate messages from TTCN-3 format into SUT format and vice versa. The implementation of the adapter and of the CoDec is based on HAPI library [13].

5 Conclusions

We described the design of a test platform for interoperability testing of healthcare applications. The test message types, test configurations and test behaviours are automatically generated out of HL7 and IHE standards. Our work is the first attempt for building a test platform for HL7 and IHE profiles based on TTCN-3 standardised test language. This framework enables testers to create reference tests which can be used for certification or validation. The design of the test system allows for flexibility and extendability by dynamically adaptation to the test configuration changes.

Currently, only PCD profile is supported but we plan extensions to other profiles and considerations of issues such as security and risk management.

References

[1] European Telecommunications Standards Institute (ETSI) ES 201 873-1 V3.2.1, *The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language*, 2007.

[2] European Telecommunications Standards Institute (ETSI) (ES) 201 873-5 V3.2.1, *The Testing and Test Control Notation version 3; Part 5: TTCN-3 Runtime Interface (TRI)*, 2007.

[3] American National Standards Institute (ANSI), *Health Level 7 (HL7) Standard*, `www.hl7.org`, 2008.

[4] Integrating the Healthcare Enterprise (IHE), `www.ihe.net`, 2008.

[5] Patient Care Devices (PCD) Technical Framework, `www.ihe.net/Technical_Framework/index.cfm#pcd`

[6] TestNGMed Project, *Testing of Next Generation Medical System*, www.testngmed.org, 2007-2009.

[7] Testing Technologies, *TTworkbench - an Eclipse based TTCN-3 IDE*, `www.testingtech.com/products/ttworkbench.php`, 2009.

[8] Applied Biosignals GmbH, *Polybench*, `www.appliedbiosignals.com/products.php`, 2009.

[9] sepp.med GmbH, *.getmore: UML based Test Generator*, `www.seppmed.de/getmore.65.0.html`, 2009.

[10] LAIKA Project: *An open source electronic health record (EHR) testing framework*, `www.laika.sourceforge.net`, 2009.

[11] R. Sahay, W. Akhtar and R. Fox, *PPEPR: plug and play electronic patient records*. In Proceedings of the 2008 ACM Symposium on Applied Computing (Fortaleza, Ceara, Brazil, March 16 - 20, 2008). SAC '08. ACM, New York, NY, 2298-2304.

[12] D. Vega, I. Schieferdecker and G. Din, *A TTCN-3 based Test Automation Framework for HL7-based Applications and Components*. In Proceeding of the Conference on Quality Engineering in Software Technology (CONQUEST) 2008. Potsdam, Germany.

[13] HAPI Project, *HL7 Application Programming Interface*, `http://hl7api.sourceforge.net/`, 2009.