2010 Second International Conference on eHealth, Telemedicine, and Social Medicine

Design of a Test Framework for Automated Interoperability Testing of Healthcare Information Systems

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

***Abstract*—One of the challenges of evolving Healthcare Information Systems used in healthcare domain is their interoperability. Interoperability not only permits healthcare institutes to take advantage of using heterogeneous solutions from different vendors but it also offers the basis for creating a larger range of services at the edge of medicine/healthcare and information technology. New services can be created in a consistent way based on common approaches of structuring and representing healthcare data. This paper presents a test framework for testing interoperability of eHealth systems. As a major result, the test framework is designed for test automation and extendability with respect to test configurations and test cases.**

***Keywords*-eHealth, HIS, HL7, Testing, TTCN-3**

I. Introduction

Quality assurance of today's medical information systems is motivated by the rapid increasing complexity of products and processes, heterogeneous architectures and limited budgets. Little attention has been given to a common methodology for assuring the quality of Healthcare Information Systems (HIS) systems by means of testing. Unifying test procedures and realising an intelligent test design adaptable to different configurations and to various equipments, are real challenges along the testing process of HIS systems. We address these challenges by developing a test methodology including a test framework based on the standardised test language TTCN-3 [1] to automate the testing process in the healthcare area.

The development of Electronic Medical Record (EMR) solutions represents the base of the information systems in the medical industry. Many vendors provide solutions which are rather provider centred approaches (i.e. proprietary protocols and message formats), interoperability not being concerned. Obviously, automated test systems can help vendors of medical devices and EMR software to validate the specification conformance and verify their functionality.

The major problem of EMR solutions, as stated in [14], is the lack of product interoperability. Many medical information systems today store clinical information about patients in proprietary formats. The adoption of standards such as Health Level 7 (HL7) [3] used for data representation, or as Integrating Healthcare Enterprises (IHE) [6] for describing interactions between actors, is an important step in enabling interoperable healthcare systems.

The problem we are trying to solve is how to 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 in order to check if it satisfy 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 approach and the results presented in this paper represent the outcome of the research project Test Automation for the Next Generation of Medical Systems (TestNGMed) [10]. The concepts presented in this paper regard the contribution of the authors to the project. The general target of TestNGMed was to develop a TTCN-3 based testing methodology for HL7 based medical systems and their components. The intended result is a test framework providing prefabricated test configurations, test data and test procedures which will enable an easy, flexible, reusable and scalable testing of HL7-based medical systems.

The rest of the paper is structured as follows. The next section provides an overview of the related work to interoperability of HL7 based healthcare applications. Section III introduces the Patient Care Device (PCD) IHE profile, which is the case study we selected to demonstrate our approach. In Section IV, we introduce the TTCN-3 based test framework. Section V briefly describes the mapping of HL7 data structure to TTCN-3 data types. Details about the generation of test configurations and test behaviours are given in Sections VI and VII. Section VIII presents some details about the adapter layer. The last section finalises the paper with our conclusions.

II. Related Work

The interoperability of HIS was not very often concerned until now. The existent testing solutions are rather in-house test tools instead of neutral, certified tools. To address the interoperability testing in a scientific way, the evaluation and selection of interoperability scenarios are necessary [15]. There is a wide range of standards concerning the integration and interoperability of medical applications to facilitate document exchange. HL7 standard defines a common message structuring scheme for all messages used in medical systems. IHE introduces profiles enabling laboratories within healthcare institutions as well as standalone laboratories to share their patient data. These standards are the basis for the interoperability testing.

Test solutions for interoperability are currently under research or development. The solution provided in [16] is designed to verify the input and output of EHR data against the standards and criteria identified by the Certification Commission for Healthcare Information Technology (CCHIT). A similar approach is proposed in [17]. 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 system under test (SUT) needs to interact with. This way, the interoperability is always tested against a reference implementation.

In order to have a full interoperability check, the methods proposed in [16] or [17] still have to be applied. Therefore, our approach should not be considered an alternative interoperability approach but rather an initial interoperability check. 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. The TTCN-3 testing language is a text-based language, which, for testers, is easy to learn and to use. It has been specifically designed for testing and inherits the most important typical programming language artifacts, but also includes important features required for the specification of tests.

III. IHE PCD Case Study

Integrating Healthcare Enterprise (IHE) [6] standardised a set of profiles organised by domains such as cardiology, laboratory, radiology, patient care devices, etc. These interaction profiles can be seen as an agreement between industry partners by means of eliminating the potential ambiguities that the HL7 standard allows for implementation. They offer developers a clear implementation path for communication standards supported by industry partners.

A profile consists of *actors*, which are abstract representations of the involved units, and a set of *transactions* which define the communication between the actors.

The design concepts and the implementation of our test system (TS) have been applied to the Patient Care Device (PCD) [7] case study. IHE Patient Care Device (PCD) [7] area addresses the integration of medical devices in the healthcare enterprises, which could lead to significant improvements in patient safety and quality of care. PCD profiles standardise the communication scenarios and flows between medical devices directly connected to a monitored care unit, for instance ventilators, blood pressure sensors, etc., and all other units from the medical environment like medical clinics, doctors, etc., interested and involved in receiving data from those medical devices.

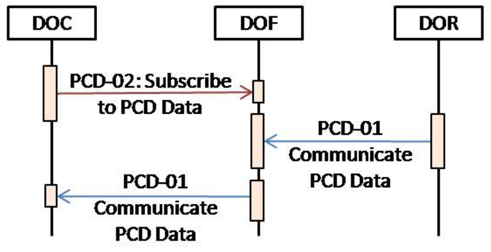
DOC
DOF
DOR
PCD-02: Subscribe to PCD Data
PCD-01 Communicate PCD Data
PCD-01 Communicate PCD Data

Figure 1. IHE PCD Profile

A. PCD Actors

Figure 1 presents 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 DOCs subscribers the possibility to receive a subset of the data stream, according to their needs and subscription predicates.

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.

B. PCD Transactions

There is a limited number of possible interactions, named *transactions*, between the actors. PCD profile standardises two transaction types which define the message structure and the message exchange flows based on the HL7 standard. However, PCD transactions impose a set of message restrictions that are clearly defined in the IHE PCD Profile.

Transaction PCD-02: Subscribe to PCD data. This transaction is used by a DOC to subscribe to a DOF for PCD data. The role of the DOC is to make a subscribe request to the DOF, containing a so-called predicate which consists of a list of attributes, which the consumer is interested in. The role of the DOF is to receive the subscribe requests, and to set up adequate filters, such that only those messages from the DOR which satisfy the filter predicates are communicated to the DOC. Each DOF is capable of supporting one or more subscribe requests from a DOC and also more than one DOC at a time.

Transaction PCD-01: Communicate PCD data. The PCD-01 message type is used to communicate PCD data from a DOR to a DOC. The role of the DOR is to transmit all the data it receives from medical devices to the DOF. The role of the DOF is to further transmit the data it receives from the DOR to the DOCs that had previously successfully subscribed to the filter. The role of the DOC is to receive the data from DOF.

As Figure 1 shows, the PCD-01 transaction type is used for the communication between DOR and DOF actors, as well as between DOF and DOC actors. A DOC can run multiple parallel transactions requesting different data sets. The duration of a PCD-01 transaction can hold from several seconds up to days or weeks according to the DOC request.

C. Interoperability Test Objectives

Starting with the interaction diagram in Figure 1, the following interoperability test objectives can be considered:

- consumer subscription: validate a subscription transaction with valid/invalid consumer ID
- consumer receives data: validate that after the subscription, the consumer receives data from the filter according to the subscription predicate; the filter should forward data received from the reporter
- consumer unsubscription: validate a unsubscription transaction with valid/invalid consumer ID
- reporter data sending frequency: validate that the reporter sends data to the filter with the correct frequency

These interactions can be combined in more complex sequences. Figure 2 shows such a test scenario which is used in the following sections to explain the test harness. A consumer with a valid ID subscribes to the filter with a predicate then waits for some data from the filter that matched that predicate; next the consumer unsubscribes. This test scenario may uncover flaws in the filter implementation with respect to consumers' subscription.

Additionaly, security test objectives can be taken into consideration. However, they were not considered so far in our setup. Testing if the consumer authenticates to an authentication agent before it subscribes to the filter, is an example of a security test.

IV. The TTCN-3 Test Framework

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

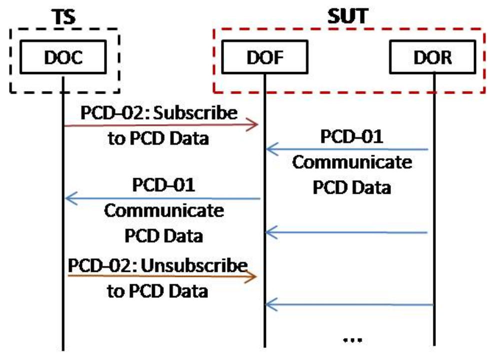
TS
SUT
DOC
DOF
DOR
PCD-02: Subscribe to PCD Data
PCD-01 Communicate PCD Data
PCD-01 Communicate PCD Data
PCD-02: Unsubscribe to PCD Data
...

Figure 2. DOC Subscription-Receive Data-Unsubscription Testcase

A TTCN-3 based test specification is called Abstract Test Specification (ATS). It usually consists of various TTCN-3 modules. A module is the top level element of the TTCN-3 language which is used to structure the test definitions of:

- *test data*: the structure of test data is defined as types of messages, and the concrete test data is represented by instances of these types called *templates*
- *test configurations*: ports and test components are used to define the active entities of the test system and the interfaces to the SUT
- *test behaviour*: *functions*, *altsteps*, and *testcases* which implement the interactions between the test components and the SUT
- *control*: the global behaviour of the test system to control the flow of test case execution

In order to get an executable test system based on TTCN-3, in addition to the abstract test specification, an *adaptation component* [2] called **Adapter** and an *encoding/decoding component* called **CoDec** need to be provided. These components are used by the execution environment (we used TTworkbench [11]), 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 Polibench tool [12], 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 [11] IDE was used.

V. HL7 to TTCN-3 Data Type Mapping

An TTCN-3 Abstract Test Specification (ATS) requires a type system that corresponds to the input and output data types of the SUT interface. Given the different versions of HL7 standard (e.g., 2.x, 3.x), the mapping of HL7 to TTCN-3 becomes more difficult to use and to adapt. The HL7 versioning problem and interoperability issues do not exist anymore in the case of SUTs compliant with IHE profiles thanks to the introduction of a set of constraints at the HL7 data structure level. These constrains propose a unified usage of HL7 data structure independent of HL7 versions. Therefore, we address only the mapping of HL7 messages defined by IHE profiles. Our work on the conceptual mapping and its realization has been presented in detail in [18].

The mapping rules from HL7 types to TTCN-3 data types follow closely the structure of the HL7 message format. A HL7 message has a tree-like structure as shown in the Figure 3 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.

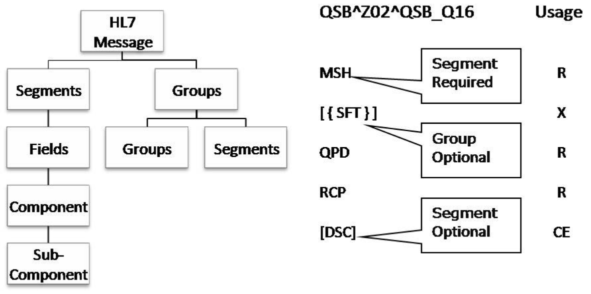

Figure 3. HL7 General Message Structure (left) and PCD-02 Message Structure (right)

In the case of the PCD profile [7], the *PCD-02* HL7 data type, used for describing a subscription message, corresponds to the HL7 message type described in the standard as *QSB^Z02^QSB_Q16* and has the structure depicted in Figure 3. The PCD profile imposes various restrictions to the generic *QSB^Z02^QSB_Q16* HL7 message type, such as, for instance, the second segment *[{STF}]*, which is optional and can infinitely repeat, will never be present in a PCD-02 message instance.

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.

As long as the test specification contains thousands of definitions spread over 10.000 TTCN-3 LOC only for covering the PCD profile, it is extremely important to be consistent in writing TTCN-3 test definitions and in maintaining a clear test suite and module structure. To keep the test specifications consistent, readable, reusable and well structured, we developed and applied a set of guidelines for writing and structuring the TTCN-3 specifications. A general approach for defining guidelines for TTCN-3 test specifications has been proposed in [19].

Listing 1. TTCN-3 and PCD-02 Message Structure

```
/*QSB^Z02^QSB^Q16*/
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,
  ST32 Query_Tag optional,
  CX_20Group_Type MRN optional,
  Pharmacy_Order_Types_Table Action_Code optional,
  PL_20Group_Type Patient_Location optional,
  CE_6Group_Type Device_Class optional,
  CE_6Group_Type Parameter_Class optional,
  TS Start_Date_Time optional,
  TS End_Date_Time optional,
  CQ Interval optional };
type record length (1..20) of CX CX_20Group_Type;
```

VI. Generation of TTCN-3 Test Configurations

The test configurations (ports, components and their parameters) have been generated out of test scenarios designed in UML.

Multiple scenarios may correspond to the same IHE profile. For example, the following scenario can be considered for the PCD profile: a consumer (DOC) successfully subscribes to the filter (DOF) and after 10 minutes of receiving patient data it unsubscribes. Another scenario may involve two consumers (DOCs) subscribing for data to the filter (DOF) in parallel and after some time both unsubscribe. Consequently, an IHE profile specification translates into many test scenarios. In order to have the TTCN-3 test scripts generated, the existence of the test scenarios is a prerequisite.

The solution that has been adopted within the project [10] is based on generating test scenarios expressed in UML Sequence Diagrams (UML-SD) out of UML Activity Diagrams (UML-AD) corresponding to an IHE profile. The TTCN-3 test scripts are next generated out of test scenarios. The technical realisation of the generation algorithms proposed in the project has been achieved within the .getmore tool [13].

A. Generation of Configuration Parameters

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*. Their values change according to the concrete configuration. Listing 2 shows which module parameters are required for testing the DOF actor. The test system needs only for the SUT actors a complete address (line 2). The DOC actors that are simulated by the test system can run at any location of choice where the test system is started.

The naming convention of module parameter identifiers plays an important role in the test configuration design. For example, the identifier *portPTC_Consumer_PRESSURE* will always be associated to one DOC actor that the test system wants to simulate.

Listing 2. TTCN-3 Module Parameters
```
modulepar {
 charstring add_SUT_Filter:="127.0.0.1:10000";
 charstring portPTC_Consumer_PRESSURE:="15001";
 charstring portPTC_Consumer_VOLUME:="15002";
}
```

B. Generation of Test Components

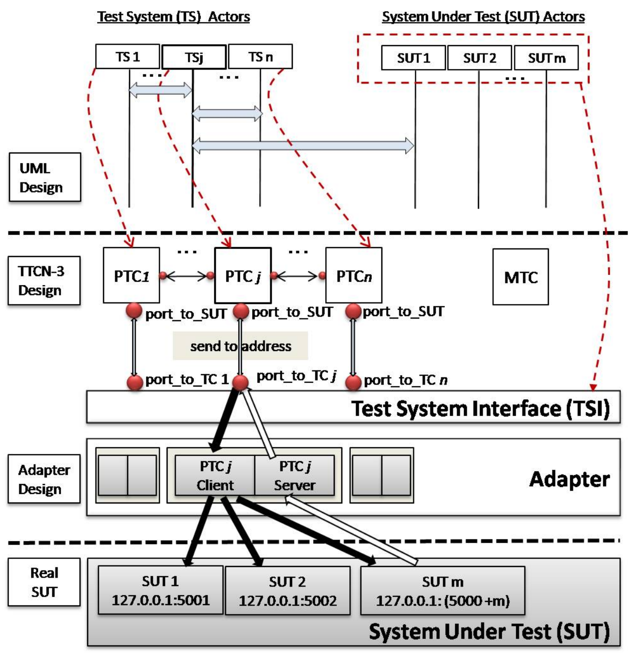

Figure 4. Test Scenario mapping to TTCN-3 Test Configurations

The test configuration design impacts the overall test system by means of performance, flexibility, extendability or self-adaptation to new test scenarios. Figure 4 illustrates the generation of TTCN-3 elements (TTCN-3 Design) out of test scenario elements (UML Design). In the following we explain these mapping rules.

A TTCN-3 test case runs on top of a *main test component* (MTC) that can create other *parallel 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). In the generation algorithm we defined the following mapping rules. 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 Consumer_PRESSURE_Comp_Type {
  port HL7Port_Type pt_data_HL7_SuT;
} type component TSI {
  port HL7Port_Type pt_data_HL7_Consumer_PRESSURE;
} type component MTC { }
```

The differentiation between various SUT entities within a test behaviour is made by using the construct *send to address*, where address is substituted by the value of the TTCN-3 module parameter corresponding to the appropriate SUT entity.

C. Generation of Communication Ports

The IHE profile specification regarding how actors interact one with each other is essential in deriving test behaviours. In the following, we present the generation rules we applied.

1) TTCN-3 Port Instances on Component Types: 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 4. 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, which requires the address of the SUT actor (e.g. 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 4, the TSI component has one port to each PTC called *port_to_TC*.

Listing 4. TTCN-3 Test Configuration - Port Type
```
type port MLLPPort_Type message {
 inout all
};
```

2) Generation of the TTCN-3 Port Types: The IHE profiles specify if the interactions between its actors occur over only one transport protocol, such as MLLP [5], or over more protocols, e.g. MLLP, DICOM [4], SOAP, etc. This information has to be encapsulated also in the test scenario, for instance, by annotating the interactions in the UML-SD with the corresponding transport protocol. In order to enable a multi-adaptation in our design, we define a TTCN-3 port type for each transport protocol that the IHE profile specifies. This port is instantiated by any PTC interacting with the SUT. The PCD IHE profile specifies only one transport protocol for transporting HL7 data, hence the TTCN-3 test configuration will contain only one port definition (Listing 4).

VII. Generation of TTCN-3 Behaviour

The behaviour of one actor is determined by the sequence of interactions it has with the other actors within the IHE profile. A UML test scenario is mapped to a TTCN-3 test behaviour and to a test configuration (see Section VI). On the test system side, this involves parallel behaviours of many (SUT and TS) actors. In Figure 4, the MTC component instantiates the PTC components. The instructions to create the PTCs are also generated from the UML test scenario. Each PTC is next started with a test behaviour which instructs that PTC to act as the actor that it simulates. Listing 5 shows an example of a TTCN-3 behaviour function corresponding to a DOC actor. It sends a subscribe message (line 10) to the DOF with a request for patient data (template defined at lines 5-8). This template is created from another predefined template by using the *modifies* construct. At line 13 a timer is started. The *alt* construct starting at line 14 is used to handle the possible reactions of the SUT. Line 15 describes the alternative when the subscription succeeded and the PCD received one PCD-01 message. The second alternative, at line 18, 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 25 for unsubscription request. The unsubscription template is defined by setting the *Action_Code* to value "D" (line 29).

Listing 5. TTCN-3 Behaviour Function Example
```
 1 function Consumer_PRESSURE_Behaviour()
 2 runs on Consumer_PRESSURE_Comp_Type {
 3  timer replyTimer;
 4
 5  template PCD_02_Message_Type modifiedTemplate_1
 6    modifies PCD_02_Send_Template_REG_PRESSURE:={
 7    MSH_Segment:={Sending_Application:={
 8              Universal_ID:="0000010000000C01"}}};
 9
10  pt_data_HL7_SuT.send(modifiedTemplate_1)
11  to address_Filter;
12
13  replyTimer.start(20.0);
14  alt {
15          [] pt_data_HL7_SuT.
16          receive(PCD_01_Receive_Template_DOF){
17                          replyTimer.stop;}
18          [] pt_data_HL7_SuT.receive{
19                  replyTimer.stop; setverdict(fail);
20                  mtc.stop; }
21          [] replyTimer.timeout{
22                  setverdict (fail); mtc.stop;}
23  }
24
25  template PCD_02_Message_Type modifiedTemplate_2
26    modifies PCD_02_Send_Template_REG_PRESSURE := {
27    MSH_Segment:={Sending_Application:={
28              Universal_ID:="0000010000000C01"}},
29    QPD_Segment:={Action_Code:="D"} };
30
31  pt_data_HL7_SuT.send(modifiedTemplate_2)
32  to address_Filter;
33 }
```

The example in Figure 2 is mapped to the testcase presented in Listing 6. The testcase creates a test configuration with a consumer component and a system component. The ports of the two components are connected using *map* statement. The consumer component is started with *Consumer_PRESSURE_Behaviour*.

Listing 6. TTCN-3 Testcase Example
```
testcase testSubRecDataUnsub()
runs on MTC system TSI  {
  var Consumer_PRESSURE_Comp_Type v_Consumer :=
  Consumer_PRESSURE_Comp_Type.create(
      "Consumer_PRESSURE" & "_" &
      portPTC_Consumer_PRESSURE);
  map(v_Consumer:pt_data_HL7_SuT,
      system:pt_data_HL7_Consumer_PRESSURE);
  v_Consumer.start(
      Consumer_PRESSURE_Behaviour());
   all component.done;
}
```

VIII. Implementation of the Adapter and CoDec

The setting we used in our project requires the usage of the MLLP [5] transport protocol for exchanging HL7 data. The adapter implementation (see Section IV) needs to support this transport layer.

For handling MLLP transactions between TS simulated actors and the SUT actors, the adapter uses a Java free implementation library for HL7 called HAPI [21].

Figure 4 illustrates how the TTCN-3 ports (TTCN-3 Design) relate to adapter components (Adapter Design). For each actor simulated by the test system, the adapter defines a pair of *server* and *client* components. The *client component* becomes active whenever the test system stimulates the SUT (e.g., a DOC subscription message) and has the additional role of dispatching the message to the desired SUT actor address. The *server component* is created and started for each TTCN-3 PTC that has direct interaction with at least one SUT actor. This component is responsible for handling connections from SUT actors. The main advantage of this design is that, whenever the test configuration changes, i.e., the number of PTCs, number of SUT actors and their interactions, the adapter does not need to be changed.

The problematic of how the design of the adapter influences and corresponds to specific needs of the test configuration, has been investigated in the context of performance testing in [20].

The role of the CoDec is to translate messages from TTCN-3 format into SUT format and vice versa. The implementation of the CoDec is also based on HL7 library

[21] that acts as a parser for HL7 messages. We developed a core library of functions that help to encode/decode HL7 basic types. These sub-types encoding/decoding functions can be used in the composition of HL7 messages required by other IHE profiles, hence increasing the extendability of the CoDec.

IX. Conclusions

In this paper, we describe the design of a test platform based on TTCN-3 standard 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 a standardised test language. This should enable testers to create reference tests which can be used for certification or validation. We also increased the level of automation by designing a flexible test configuration mechanism which maps the IHE profile actors to TTCN-3 test components and ports.

Moreover, the design of the adaptation layer allows for more flexibility and generality. The adapter recognises the dynamic changes of the test configuration and automatically adapts to them. The adapter also detects automatically which protocols should be used for communication (as required by the IHE profile).

Currently, only PCD profile is supported; further development plans include extensions to other profiles and considerations of issues such as security and risk based testing.

References

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

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

[3] American National Standards Institute (ANSI), *Health Level 7 (HL7) Standard*, www.hl7.org. Accessed 4 Dec. 2009.

[4] NEMA, *Digital Imaging and Communications in Medicine (DICOM)*, http://medical.nema.org/. Accessed 4 Dec. 2009.

[5] SUN MLLP, *Minimal Lower Layer Protocol Release (MLLP)*, www.hl7.org/v3ballot/html/infrastructure/transport/transport-mllp.htm, Accessed 4 Dec. 2009.

[6] Integrating the Healthcare Enterprise (IHE), www.ihe.net. Accessed 4 Dec. 2009.

[7] Patient Care Devices (PCD) Technical Framework, www.ihe.net/Technical_Framework/index.cfm#pcd. Accessed 4 Dec. 2009.

[8] National Electrical Manufacturers Association (NEMA), *Digital Imaging and Communications in Medicine (DICOM)*, http://medical.nema.org. Accessed 4 Dec. 2009.

[9] ReTeMes Project, *Reliability Testing Of Medical Systems (E!4053-RETEMES)*, http://www.eureka.be/inaction/AcShowProject.do;jsessionid=7f0000011f900f0b8d972e96440cbb3ec143ab2a632a?id=4053. Accessed 4 Dec. 2009.

[10] TestNGMed Project, *Testing of Next Generation Medical System*, 2007-2009 www.testngmed.org. Accessed 4 Dec. 2009.

[11] Testing Technologies, *TTworkbench - an Eclipse based TTCN-3 IDE*, www.testingtech.com/products/ttworkbench.php. Accessed 4 Dec. 2009.

[12] Applied Biosignals GmbH, *Polibench*, www.appliedbiosignals.com/products.php. Accessed 4 Dec. 2009.

[13] sepp.med GmbH, *.getmore: UML based Test Generator*, www.seppmed.de/getmore.65.0.html. Accessed 4 Dec. 2009.

[14] M. Eichelberg, T. Aden, J. Riesmeier, A. Dogac and G. B. Laleci, *A survey and analysis of Electronic Healthcare Record standards*, ACM Comput. Surv. Vol. 37, Nr. 4, pages 277-315, Dec, 2005.

[15] J. A. Mykkänen and M. P. Tuomainen, *An evaluation and selection framework for interoperability standards*, Inf. Softw. Technol. Vol. 50, Nr. 3, pages 176-197. Butterworth-Heinemann. Feb, 2008.

[16] LAIKA Project: *An open source electronic health record (EHR) testing framework*, www.laika.sourceforge.net. Accessed 4 Dec. 2009.

[17] 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.

[18] 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.

[19] G. Din, D. Vega and I. Schieferdecker, *Automated Maintainability of TTCN-3 Test Suites based on Guideline Checking*. The 6th IFIP Workshop on Software Technologies for Future Embedded & Ubiquitous Systems (SEUS 2008), Springer, October 2008.

[20] G. Din, *An IMS Performance Benchmark Implementation based on the TTCN-3 Language*. STTT Journal, Vol. 10, Nr. 4, pages 359-370, 2008.

[21] HAPI Project: *HL7 Application Programming Interface*, http://hl7api.sourceforge.net/. Accessed 4 Dec. 2009.