A CONFORMANCE AND INTEROPERABILITY TEST SUITE FOR TURKEY'S NHIS AND AN INTERACTIVE TEST CONTROL AND MONITORING ENVIRONMENT

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ALİ ANIL SINACI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

JUNE 2009

Approval of the thesis:

A CONFORMANCE AND INTEROPERABILITY TEST SUITE FOR TURKEY'S NHIS AND AN INTERACTIVE TEST CONTROL AND MONITORING ENVIRONMENT

submitted by **ALİ ANIL SINACI** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering , Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, **Graduate School of Natural and Applied Sciences** _____

Prof. Dr. Müslim Bozyiğit
Head of Department, **Computer Engineering** _____

Prof. Dr. Asuman Doğaç
Supervisor, **Department of Computer Engineering, METU** _____

Assoc. Prof. Dr. Ahmet Coşar
Co-supervisor, **Department of Computer Engineering, METU** _____

Examining Committee Members:

Prof. Dr. İsmail Hakkı Toroslu
Department of Computer Engineering, METU _____

Prof. Dr. Asuman Doğaç
Department of Computer Engineering, METU _____

Prof. Dr. Özgür Ulusoy
Department of Computer Engineering, Bilkent University _____

Assoc. Prof. Dr. Ahmet Coşar
Department of Computer Engineering, METU _____

Yıldıray Kabak
SRDC Ltd. _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: ALİ ANIL SINACI

Signature :

ABSTRACT

A CONFORMANCE AND INTEROPERABILITY TEST SUITE FOR TURKEY'S NHIS AND AN INTERACTIVE TEST CONTROL AND MONITORING ENVIRONMENT

Sınacı, Ali Anıl
M.S., Department of Computer Engineering
Supervisor : Prof. Dr. Asuman Doğaç
Co-Supervisor : Assoc. Prof. Dr. Ahmet Coşar

June 2009, 118 pages

Conformance to standards and interoperability is a major challenge of today's applications in all domains. Several standards have been developed and some are still under development to address the various layers in the interoperability stack. Conformance and interoperability testing involves checking whether the applications conform to the standards so that they can interoperate with other conformant systems. Only through testing, correct information exchange among applications can be guaranteed. National Health Information System (NHIS) of Turkey aims to provide a nation-wide infrastructure for sharing Electronic Health Records (EHRs). In order to guarantee the interoperability, the Ministry of Health (MoH), Turkey, developed an Implementation/Integration/Interoperability Profile based on HL7 standards. Test-BATN - Testing Business Process, Application, Transport and Network Layers - is a domain and standards independent set of tools which can be used to test all of the layers of the interoperability stack, namely, the Communication Layer, Document Content Layer and the Business Process Layer.

In this thesis, the requirements for conformance and interoperability testing of the NHIS are

analyzed, a testing approach is designated, test cases for several NHIS services are developed and deployed, and a test execution control and monitoring environment within TestBATN is designed and implemented through the identified testing requirements. The work presented in this thesis is part of the TestBATN system supported by the TÜBİTAK TEYDEB Project No: 7070191 in addition by the Ministry of Health, Turkey.

Keywords: conformance testing, interoperability testing, National Health Information System of Turkey, test suites, automated human-driven testing

ÖZ

TÜRKİYE ULUSAL SAĞLIK BİLGİ SİSTEMİ'NİN UYGUNLUK VE BİRLİKTE İŞLERLİK TESTLERİ VE İNTERAKTİF TEST KONTROL VE İZLEME ORTAMI

Sınacı, Ali Anıl
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Asuman Doğaç
Ortak Tez Yöneticisi : Doç. Dr. Ahmet Coşar

Haziran 2009, 118 sayfa

Standartlara uygunluk ve birlikte işlerlik, günümüz uygulamaları için büyük önem taşımaktadır. Birlikte işlerlik katmanlarının farklı bölümleri için geliştirilen birçok standart bulunur. Uygunluk ve birlikte işlerlik testleri, standartlara uygunluğun ve standartlara uygun uygulamaların birlikte çalışıp çalışamadıklarının testlerini içerir. Uygulamalar arasındaki iletişimin doğruluğu ancak test vasıtasıyla garanti edilebilir. Türkiye'nin Ulusal Sağlık Bilgi Sistemi (USBS), Elektronik Sağlık Kayıtları'nın ulusal ölçekte paylaşılmasını amaçlayan bir altyapı sunar. Bu altyapı çerçevesinde geliştirilmekte olan uygulamaların birlikte işlerliğini sağlamak amacıyla, T.C. Sağlık Bakanlığı HL7 standartlarına dayalı bir geliştirme/entegrasyon/birlikte işlerlik profili oluşturmuştur. TestBATN, uygulamaların tanım kümelerinden ve kullanılan standartlardan bağımsız olarak çalışan, birlikte işlerlik katmanlarının tümünü - Haberleşme Katmanı, Doküman İçerik Katmanı ve İş Süreci Katmanı - test etme yetisine sahip bir sistemler bütünüdür.

Bu tez çalışmasında, USBS kapsamındaki uygulamaların uygunluk ve birlikte işlerlik testleri için gereksinim analizi yapılarak bir test metodolojisi geliştirilmiştir. Belirlenen metodoloji

üzerinden birçok USBS servisi için test senaryoları geliştirilmiş ve TestBATN dahilinde bir test kontrol ve izleme ortamı tasarlanmış ve hayata geçirilmiştir. Sunulan altyapı ve geliştirilen test senaryoları, USBS'deki uygulamaların uygunluk ve birlikte işlerlik testlerinde kullanılmış ve ulusal çaptaki entegrasyona büyük katkı sağlamıştır. Bu çalışma TÜBİTAK - TEYDEB, 1507 kapsamındaki B.02.1.TBT.0.06.02.162.01 Sayılı ve "Birlikte İşlerlik Standartlarının İnternet Tabanlı Test Altyapısı-7070191" başlıklı proje dahilinde geliştirilen TestBATN sisteminin parçasıdır.

Anahtar Kelimeler: uygunluk testleri, birlikte işlerlik testleri, Türkiye Ulusal Sağlık Bilgi Sistemi, Sağlık-Net, etkileşimli test kümeleri

To my family

ACKNOWLEDGMENTS

I would like to express my sincere gratitude and appreciation to my supervisor, Prof. Dr. Asuman Doğaç for her encouragement, guidance and support throughout this study.

I would also like to express gratitude to my co-supervisor, Assoc. Prof. Dr. Ahmet Coşar for his guidance and support during my study.

I am deeply grateful to my family for their love and support. Without them, this work could not have been completed.

I am deeply grateful to Tuncay Namlı without whose invaluable guidance and contribution, this work could not have been accomplished. I am also deeply thankful to Güneş Aluç for his suggestions and continuous support in the development of the testcases. I am highly indebted to my friends, Gökçe Banu Laleci Ertürkmen, Mustafa Yüksel, Yiğit Boyar, Atasay Gökkaya and all the other colleagues at the Software Research and Development Center, whose help, stimulating suggestions and encouragement helped me at all times in this research.

I would like to thank the Scientific and Technological Research Council of Turkey (TÜBİTAK) for providing the financial means throughout this study.

Finally, my special thanks go to my friends Andaç, Birkal, Ferhat, Goncagül, Hasan Şevki, Özgehan, Şerife and Zülfükar for their help, support and cheerful presence through the course of this study. Thanks for giving me a shoulder to lean on whenever I need.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

AUT	Actor Under Test
CDA	Clinical Document Architecture
EHR	Electronic Health Record
ETSI	European Telecommunications Standards Institute
eTSL	OASIS Event Driven Test Scripting Language
FMIS	Family Medicine Information System
HCRS	Health Coding Reference Server
HIS	Hospital Information System
HL7	Health Level Seven
MERNIS	Central Demographics Management System of Turkey
MHDS	Minimum Health Data Set
MoH	Ministry of Health, Turkey
NHDD	National Health Data Dictionary
NHIS	National Health Information System of Turkey
OID	Object Identifier
RIA	Rich Internet Application
RIM	Reference Information Model
SSL	Secure Sockets Layer
SUT	System Under Test
TDL	Test Description Language

TestBATN	Testing Business Process, Application, Transport and Network Layers
TTCN	Testing and Test Control Notation
XML	eXtensible Markup Language
XPATH	XML Path Language
XSD	XML Schema Definition

CHAPTER 1

INTRODUCTION

Today, eBusiness applications are widely adopted by the actors of several industry domains, governments and the public sector. These intensive relationships between the different applications belonging to a wide range of domains require standardization in the Communication Layer, Document Content Layer and the Business Process Layers, which together form the so-called interoperability stack. In the context of the different implementations of the interoperability stack by several different applications, it is still cumbersome to reach interoperability of the solutions and to achieve conformance to standards addressing the different layers of the stack. Therefore, the need for advanced testing methodologies and practices which cover relevant set of standards and specifications is increasing continuously.

Standardized protocols and services, and the applications developed through the protocols can be formally tested in two related but different ways. One way goes through the conformance testing. Conformance testing depicts whether an application correctly implements a particular standardized protocol or not. In other words, the testing of the conformance shows whether or not a single implementation of the protocol meets the conformance requirements specified for that protocol.

When a software instance includes an implementation of a universally standardized protocol, it becomes possible to specify the test criteria and procedures with a quality comparable to that of the protocol standards themselves. The protocols may set some rules on several layers of the interoperability stack and these rules are formally defined through some formal documentation. XML Schemas may restrict the structure of the content, business profiles may specify the choreography of the exchanged messages among the parties etc. Therefore, these formal restrictions form a basis for the testing applicability on that of the softwares which

implemented the protocols.

Interoperability is the ability of two or more systems or components to exchange information and to use the information that has been exchanged [1]. More specifically, interoperability is said to exist between two applications when one application can accept data (including data in the form of a service request) from the other and perform the task in an appropriate and satisfactory manner (as judged by the user of the receiving system) without the need for extra operator intervention [2].

The purpose of the interoperability testing is to prove that end-to-end functionality between, at least, two communicating systems is as by required by the standard or a number of standards on which those systems are based. However, one should keep in mind that interoperability tests are applied at the end-points and on the functional interfaces of the applications, that is, interoperability testing can only specify functional behavior. [3].

In the scope of this thesis, we analyzed the conformance and interoperability testing requirements of National Health Information System (NHIS) [4] through the Implementation/Integration/Interoperability Profile [5] which is published by Ministry of Health (MoH), Turkey. NHIS provides 25 HL7 [6] based Web services for the use of Family Medicine Information Systems (FMISs) and Hospital Information Systems (HISs). Throughout the thesis, a comprehensive testing methodology is designated and several testsuites and testcases are developed through the methodology and registered to the TestBATN framework to enable the conformance and interoperability testing of NHIS. Furthermore, to realize the testing process, the TestBATN Control and Monitoring Environment is designed and implemented.

Health Level Seven (HL7) [6] is a not-for-profit ANSI [7] accredited Standards Developing Organization. The main purpose of HL7 is to provide standards for the exchange of clinical and administrative data between healthcare systems. HL7 provides standards for interoperability that improve care delivery, optimize workflow, reduce ambiguity and enhance knowledge transfer among all stakeholders, including healthcare providers, government agencies, the vendor community, fellow SDOs and patients.

HL7 Clinical Document Architecture (CDA) [8], previously called Patient Record Architecture (PRA), is a document markup standard that specifies the structure and semantics of a clinical document (such as a discharge summary or progress note) for the purpose of ex-

change. A clinical document includes clinical observations and services about care events. A valid CDA document is encoded in Extensible Markup Language (XML) [9] and conforms to the CDA Schema which is derived from the CDA Hierarchical Description based on the XML Implementable Technology Specification.

Sağlık-NET is an integrated, secure, fast and extensible information and communication platform that aims to increase the efficiency and quality of the health services in Turkey by collecting the health care related information from the health care institutes through the defined standards and specifications.

National Health Information System of Turkey (NHIS) [4], which is developed under Sağlık-NET, is based on sharing a functional database which is accessible by authorized people and institutions with defined access rights that covers all the citizens' health records from the birth and throughout his/her life on a spine of communication network with high bandwidth throughout the entire country and using the technologies reaching telemedicine applications in professional practice [10].

All of the software systems running in the medical institutes in Turkey, the FMISs and HISs, are obliged to have the ability to transfer EHRs, called "Transmission Schema" instances to the NHIS servers at the Ministry of Health (MoH) premises. In order to guarantee the interoperability, the MoH, published an Implementation/Integration/Interoperability Profile [5] for FMIS and HIS vendors. The Integration Profile and its reference specifications present all the restrictions and requirements for vendors to update or develop the necessary components within their FMISs and HISs for a successful integration. However, without an extensive and effective testing process this is a difficult job for those vendors. Furthermore, only through testing, correct information exchange among these eHealth applications can be guaranteed and the products can be certified.

Conformance and interoperability testing are both important and useful to the testing of the applications within NHIS. Conformance testing of the FMISs and HISs can show that those implementations comply with the requirements of the protocols and specifications asserted in the Integration Profile of MoH.

In the scope of this thesis, a testing methodology is devised which adopts a "step-by-step" approach from the basic and simple testcases to the complex ones. Basic testcases include

test steps which require the minimum level of assertions according to the Integration Profile. That is, the FMISs and HISs can apply the basic conformance testcases successfully if they meet the minimum set of functionalities dictated by the related service's specifications.

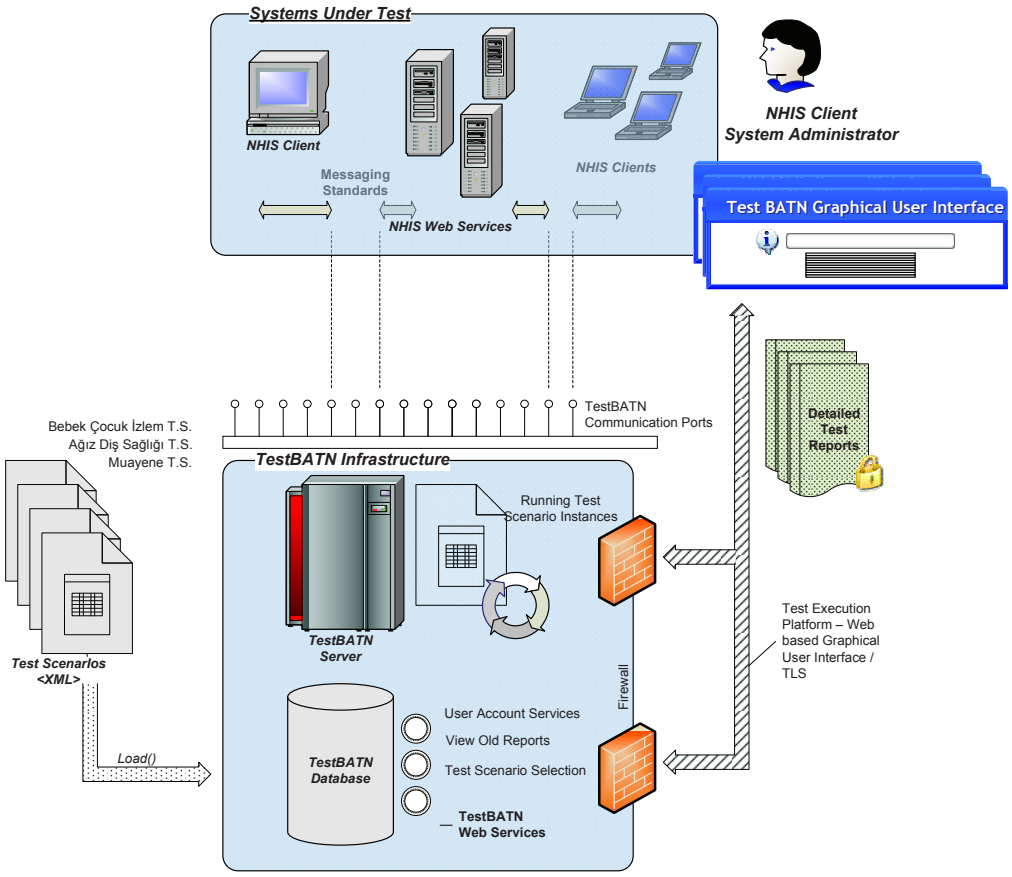

Figure 1.1: TestBATN Architecture within the scope of NHIS

As presented in Figure 1.1, several testcases are developed through the designated testing methodology and registered to the TestBATN framework. The developed testcases are published to the FMIS and HIS vendors to enable the online testing of their products. Testcases are developed in the scripting language, Test Description Language (TDL), of TestBATN and published to the online use of the clients through the TestBATN Control and Monitoring Environment, which is also a part of this thesis.

TestBATN is a software framework which proposes a design and an execution environment and applies a specific testing approach for dynamic, configurable and automated execution of

conformance and interoperability testing against B2B standards, profiles or specifications.

TestBATN is a comprehensive and integrated system consisting of several components. The TestBATN Engine is the system which interprets and executes the testcase definitions, which are scripted in the Test Description Language (TDL), and manages the whole process. The TestBATN Control and Monitoring Environment is the interface between the engine and the Human Test Driver to provide real time monitoring on the test execution, some control on the scenario and reporting of the test results. The Human Test Driver, while monitoring the test execution with TestBATN Control and Monitoring Environment, manages the SUT according to the instructions shown on the Graphical User Interfaces (GUIs).

TestBATN Control and Monitoring Environment is designed and implemented as Adobe Flex based Rich Internet Application (RIA) through the scope of this thesis to enable the online and high speed testing capabilities within the testing of NHIS. The environment also implements a Test Driving Protocol on the TCP [11] level to interact with the TestBATN Engine enabling the high speed, instant communication.

Since June, 2008, the developed testcases are in the online use of the FMIS and HIS vendors, up to 70 companies with nearly 340 users, on “http://www.srdc.com.tr/testbatn”. Since then, nearly 20.000 testcase executions are recorded. Apart from the online use of the developed testcases through TestBATN; MoH, Turkey organized two well-attended integration workshops. First one of the workshops is organized in Çeşme, İzmir during June, 29 - July, 5 2008. Nearly 130 FMIS/HIS developers from 50 vendor companies attended to Çeşme workshop and tested their client applications successfully with the developed testcases registered to the TestBATN framework.

The testcases within the TestBATN framework are also used in the second integration workshop, which is held in Ankara during December 19 - 21, 2008. 133 attendees from 64 vendor companies were there to test their FMIS/HIS implementations.

The work presented in this thesis is supported by the TÜBİTAK TEYDEB Project No: 7070191 in addition by the Ministry of Health, Turkey.

In the rest of the thesis, Chapter 2 gives detailed information about the background technologies, standards and specifications. Particular segments of NHIS and TestBATN framework, the development strategies and enabling technologies are presented in Chapter 2. Chapter 3

discusses the testing methodology adopted in the testing of NHIS. Developed testcases and example scenarios with the detailed explanations are provided also in Chapter 3. Chapter 4 elaborates the design and implementation issues of TestBATN Control and Monitoring Environment. The chapter also gives detailed description of the Test Driving Protocol constructs. Chapter 5 presents the related work in the testing context. Finally, Chapter 6 concludes the thesis and discusses the ideas about the future work.

CHAPTER 2

BACKGROUND ON ENABLING TECHNOLOGIES AND STANDARDS

2.1 Materials Used In Testing

Testing process that is being introduced in this thesis derives benefit from several XML [9] based materials/technologies. Since National Health Information System of Turkey adopts XML based standards and TestBATN also uses XML based technologies and languages, the testing approach of this thesis also follows the means of XML based technologies.

XML Schema Definition - XSD

An XML schema is a description of a type of XML documents, typically expressed in terms of constraints on the structure and content of documents of that type, above and beyond the basic syntax constraints imposed by XML itself. There are a number of different languages available for specifying an XML schema. XML Schema Definition [13] is one of those XML schema languages which provides a means for defining the structure, content and semantics of XML documents. An XSD schema dictates a set of rules to describe the structure of the XML documents. An XML document is said to be 'valid' according to the XSD schema, when the XML document conforms the rules dictated by that XSD schema. Those valid XML documents are called as instances of the schema.

XSD schemas include a set of components, namely the definitions of elements and attributes of the elements. The elements can be nested and ordered within each other. XSD schema allows definition of restrictions on the child elements of each element and their attributes. The data types for elements and attributes are also set through the XSD schemas. More-

over, default and fixed values of the elements and attributes can be defined through schema definitions.

XSD version 1.1 is currently under development by the XML Schema Working Group under the World Wide Web Consortium (W3C) [14].

XML Path Language - XPATH

XML Path Language, XPath, [15] is a query language for selecting parts of an XML document. The primary purpose of XPath is to address the nodes inside an XML document. In addition to its primary purpose, it also provides basic facilities for manipulation of primitive data types such as strings, numbers and booleans. XPath uses a compact, non-XML syntax to facilitate use of XPath within URIs and XML attribute values.

The query methodology adopted by XPath, relies on the tree structure of the XML documents as well as atomic values such as integers, strings, and booleans, and sequences that may contain both references to nodes in an XML document and atomic values. That is, XPath operates on the abstract, logical structure of an XML document, rather than its surface syntax. In this way, the XML document is processed as a tree of nodes. There are different types of nodes such as, element nodes, attribute nodes and text nodes. XPath accesses these nodes with the result of its evaluation.

XPath is based on expressions which can be constructed from keywords, symbols and operands. XPath enables the nested definition of expressions. The result of an XPath expression consists of a selection of nodes from the input XML documents without the duplicates and additional evaluation specific parameters.

There exist two versions of XPath, XPath 1.0 and XPath 2.0. Both of them are Recommendation by W3C. XPath 2.0 provides more powerful constructs than XPath 1.0, however, currently; XPath 1.0 is more widely accepted in the community.

The Schematron

The Schematron [16] [17] is an XML Schema language which defines the structure, content and semantics of XML documents by making well-defined assertions about patterns found in XML documents. The main difference of Schematron from the other schema definition languages is its basis on the tree patterns residing in the XML document model instead of

the grammars. From another perspective, Schematron is a rule based validation language over the XML documents. Namely, a Schematron file includes a set of XPath-based rules, which correspond to the assertions, to test the presence and/or absence of patterns in XML tree models.

The Schematron is comparatively stronger in the structural validation of the instance documents because of its rule-based system. It checks the patterns existing in the XML documents' tree models against the rules asserted through the Schematron language. This process includes XSL Transformations [18] one after another. Schematron also enable the error messages expressed in natural language which will come up during the validation phase. This makes error detection much easier than the cryptic error codes. On the other hand, Schematron has weaknesses against the other schema definition languages. Due to its rule-based nature, it can be very complex to specify the basic structure and content definition with a set of rules. Therefore, combining the use of the Schematron with another schema definition language, such as XSD, induces a good solution.

The Schematron is an ISO (the International Organization for Standardization) [19] standard. It has been standardized to become part of ISO/IEC ISO/IEC 19757 - Document Schema Definition Languages (DSDL) - Part 3: Rule-based validation - Schematron.

2.2 Enabling Technologies

Testing process requires use of specific software to test the operations of the client programs and utilities automatically. TestBATN constitutes a testing system behaving as an application server. NHIS clients need to communicate with TestBATN engine for the testing of their application programs. Enabling this through the Internet, based on Rich Internet Application technologies, is the main purpose of the testing approach of this thesis.

Rich Internet Applications (RIAs)

Rich Internet Applications (RIAs) are web applications, accessible through web browsers, which are carrying characteristics of the regular desktop applications. To describe the benefits of Rich Internet Applications, understanding the early internet applications plays an important role. As the computers started to be interconnected, World Wide Web came up and static

web pages evolved within the well-known client-server architectures. In these very early systems, the clients interact with the server through HTTP [20] and HTML [21]. The primary technology in creating Internet applications is HTTP, which is simple, ubiquitous, requires no special tools. The clients send HTTP requests to the application servers and the servers return static HTML pages to the clients in this model. This early static model is mostly concerned with the presentation. Since there is no dynamism and interactivity, the only process is the rendering of the HTML data and to present it to the user looking at the browser.

Dynamic web pages showed up since the static client-server model did not satisfy the changing user and system requirements. Several server side scripting languages and web frameworks evolved and enabled the dynamic creation of web pages. The examples of server-side scripting languages are PHP [22], ASP.NET [23], JSP [24], ColdFusion [25] and other languages. The dynamism of this model is also based on the HTTP and HTML technologies with synchronous communication over TCP [11]. The application server creates dynamic HTML pages upon the request coming from the clients. Dynamic creation of the web pages has brought interactivity in the interaction with the end-users. On the other hand, client-side scripting has also evolved to enhance the interactivity at the presentation level on the user web browsers. Client-side scripting languages like JavaScript [26] or ActionScript [27] are frequently used to orchestrate media types (sound, animations, changing text, etc.) of the presentation.

The Internet age evolves very fast and this evolution pushes the Internet users to demand more sophisticated and increasingly interactive web sites. Synchronous client-server model within an “ask-response” manner, relies on the server for processing, requires refreshment of content pages and communication through HTTP - HTML bundle at each user request. This, results in a lot of redundant data being transferred, increasing wait times for clients, and giving a start and stop feel within the web pages due to their multi-page interfaces. With the very successful increases in the speed of modern computers and increasing accessibility to broadband internet services, users are demanding more from web-based applications. Therefore, to meet the today’s Internet users’ requirements, there is a need for more efficient and effective communication methodologies between the user browsers and application servers. Furthermore, friendlier and more esthetic graphical interfaces are needed to satisfy the users than the HTML whose main purpose is just the presentation of data rather than the development of rich interfaces.

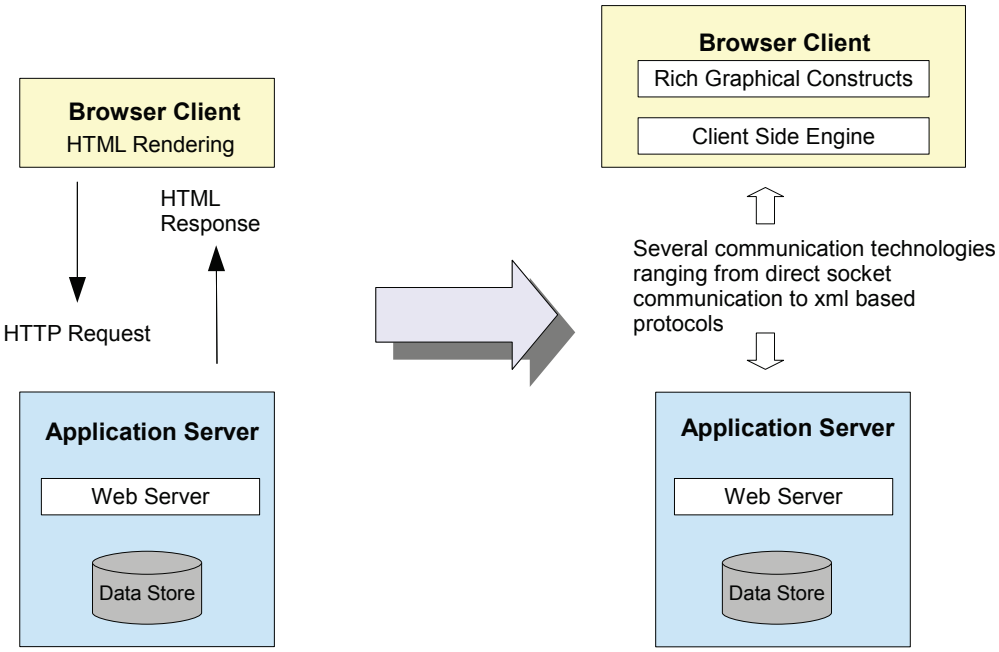

Figure 2.1: Evolution of the Web

The outcomes of the today's web application development efforts through the light of the limitations of the traditional client-server model are the Rich Internet Applications and several associated tools and frameworks. Briefly, Rich Internet Applications provide rich graphical constructs for friendlier user interfaces and adopts asynchronous methods enable real time data communication and dynamic rendering. As seen in Figure 2.1, with the evolution to richer clients, several communication alternatives have been approached and the graphical presentation layer induced richer and friendlier interfaces to the end-users. Resulting web applications give the feel of a desktop-like application to the user. A desktop application works with the information sources residing at databases and the file system at the back-end. A RIA can be seen as a desktop application working with remote databases and file systems, overcoming the limitations of the Internet communication with asynchronous and dynamic techniques. Consequently, RIAs can be seen at the intersection of Desktop Applications, Web Applications and Communication Technologies.

RIAs differ in their development and deployment processes. This, leads to a distinct categorization among the RIA based applications [28]. The first category includes the RIAs served as embedded solutions or standalone applications launched from the browser. This, the first

type of RIAs category, is also referred as the “sandbox” approach. An example of the embedded practice is the Adobe Flex [29] technology. Adobe Flex enables the creation of Flash [30] based RIAs running inside the engines plugged-in to the browsers. The resulting Flash application is embedded inside the HTML documents. Java Web Start [31] constitutes an example of the stand-alone practice of RIAs with the use of Java Network Launch Protocol (JNLP) [32]. JavaFX [33] is another example of the sandbox approach from the Java side. The sandbox approach enables a stable and easy development environment like the traditional programming environments, because the execution of the web application is handled inside a run-time at the client side and the data communication may occur within several alternatives like in a desktop environment. Presence of such run-times as plug-ins to the browsers also guarantees the multi-platform execution of the application as long as the plug-in of the web browser is capable of running the RIA.

RIAs developed with the use of Ajax [34] technologies constitute to the script based type of the RIA development and deployment categories. Using Cascading Style Sheet (CSS) [35] in combination with HTML to enrich the rendering of the pages on the screen and adoption of asynchronous communication methods available in JavaScript also realized RIA environments in some manner. This second type, script based RIA category mainly suffers from compatibility problems since there exist remarkable divergences among the different browsers, and their different versions of the same browser, in the implementation of JavaScript, DOM [36] and CSS processing. Although this type of RIA development requires no additional software installation and special run-time environment, it is a seamlessly more difficult process to proceed the development while keeping the compatibility issues in mind throughout the while process.

The third type of RIA category is browser-based. Mozilla XUL [37] is the example of this type of RIA development category. It provides an XML based language to create feature-rich, cross-platform client applications. Even though it exhibits a rich functionality options, strict dependence on the browser type limits this type of RIAs widespread use.

Apart from the categorization of RIAs, all development approaches target to the same purpose. The client should be capable of doing more job than just rendering pages. It should able to perform complex computations, send and retrieve data in the background, even at the TCP level, asynchronously upon the user requests, operate on sections of the screen and ben-

efit from the complex use of the multimedia technologies. The clients in a RIA need to be independent from the server applications to which they are connected to.

Adobe Flex

Adobe Flex is a cross-platform, open source framework for the development and deployment of rich internet applications that run identically in all major browsers and operating systems which are available today. Flex can be seen as an evolution of the earlier Flash development frameworks. Therefore, Flex applications run on the Flash run-times, those are the Flash Players [38], of the browsers. Flash Player itself is a virtual machine capable of running Flash files. This is the reason behind the well-known compatibility advantage of Flex applications.

Flex applications are the compilation of an XML-based user interface markup language, called as MXML, and an object oriented language called ActionScript directly into Flash SWF [39] binaries. MXML provides powerful constructs to create rich graphical user interfaces, manage the layout properties and dynamic facilities of the interface components. It can be seen as a tag library with support of several components which can be used in a web user interface. Apart from the user interfacing and presentation capabilities, MXML can also be used to realize complex business logic operations, data management and application behaviors. However, as presented in Figure 2.2, the tendency is to use MXML in combination with ActionScript to create RIAs.

ActionScript is an object oriented programming language to realize the application behavior and logic for those are familiar with object oriented way of programming. It is primarily used to develop Flex applications; however it is originally a scripting language based on ECMAScript [40] and has a usage on a number of other development environments.

Flex provides a data binding mechanism which makes easy to implement the Model-View-Controller pattern. Flex data binding works as mapping the value of one property of an object to a property of another object. The data needed for the application logic is kept in the model and parts of the data are bonded to the view components at the presentation layer. Upon the user or business triggers, the controller updates the data stored in the model and, throughout the Flex data binding mechanism; the presentation layer is automatically updated without additional programming efforts.

Flex supports a wide range of technologies at data services layer to enable the various com-

```
<mx:VBox width="100%" horizontalAlign="center">
  <view:HeaderView width="40%" />
  <mx:Spacer height="20" />
  <mx:ApplicationControlBar width="50%" paddingBottom="10">
    <mx:VBox width="100%">
<mx:ComboBox width="75%" id="cmbOntologies" />
<mx:HBox width="100%">
  <mx:TextInput id="txtSearch" width="100%" />
  <mx:Button label="Search" click="search()"/>
</mx:HBox>
    </mx:VBox>
  </mx:ApplicationControlBar>
</mx:VBox>

<mx:Script>
  <![CDATA[
import mx.collections.ArrayCollection;

private function search():void {

}
  ]]>
</mx:Script>
```

Figure 2.2: MXML & ActionScript sample from a Flex application

munication approaches between the clients and the application servers. It provides a rich messaging infrastructure to realize data-rich Flex applications. It enables communications through simple HTTP calls, SOAP [41] Web Service calls, low-level socket interactions and other various efficient built-in supports for the communication with Java based application servers. Furthermore, Flex framework exposes application-level services, including history management, layout management, cursor management, exception handling, internationalization, logging, and other types of services.

The Cairngorm Micro-architecture

Cairngorm [42] is lightweight micro-architecture to ease the development and maintenance of Flex based RIAs. Cairngorm provides a starting point for the technical architectures of the applications that will be built through Flex. It can be seen as a collection of small software design patterns which contribute to the adoption of the Model-View-Controller model approach inside the RIAs.

Figure 2.3 presents an explanation of the Cairngorm micro-architecture. Basically, it addresses three important points in the RIA development through Flex:

- Handling user gestures on the client: User gestures are generally represented as events

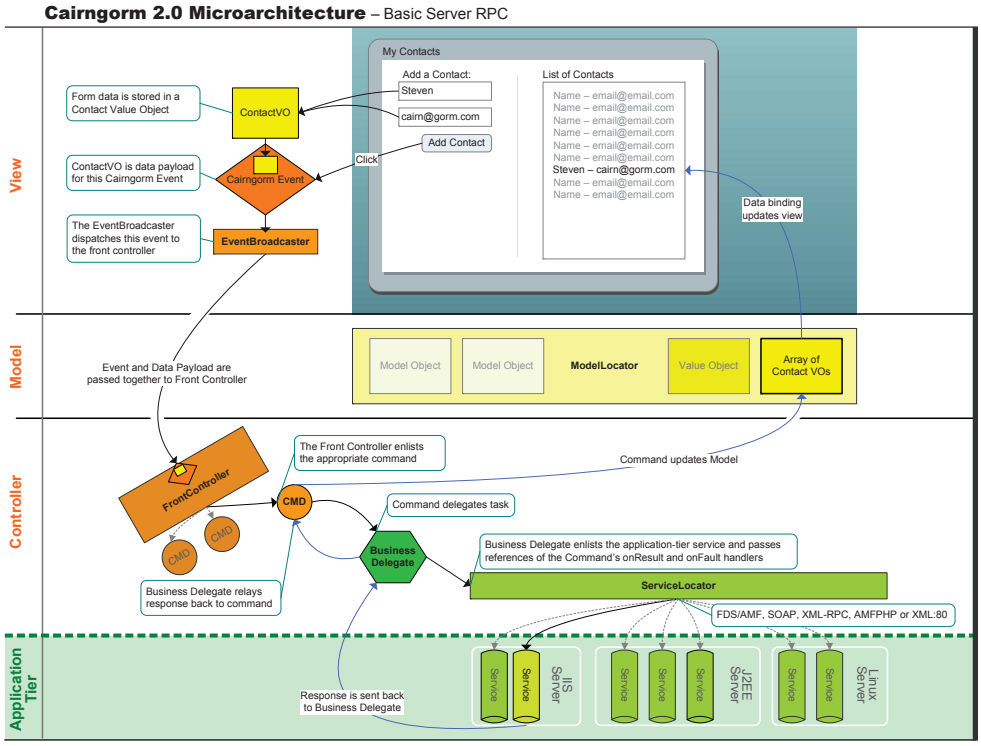

Figure 2.3: Cairngorm micro-architecture [43]

specific to the Cairngorm micro-architecture. These events are carried through a controller to the appropriate commander to realize the desired action. The commander manages the data residing in the model of the application and this is the only way to interact with the model. By this way, the model is kept consistent and stable and only edited through the special Cairngorm events.

- Encapsulating business logic and server interaction: The communication backend of the RIA with the application server is abstracted and encapsulated inside a service locator. By this means, the data service approach is integrated through well-defined interfaces. Since, Flex provides several approaches to communicate with application servers; any change in the data service capsule is done independent of the other modules and layers of the RIA.

- Managing state on the client and representing this state to the user interface: Binding - mechanism that comes built-in within Flex enables the automatic updates on the presentation views when the corresponding sources residing in the model get updated. Thus,

while keeping the state information in the data model, it can be easily reflected to the user interface throughout the application.

2.3 Base Standards and Specifications

National Health Information System (NHIS) of Turkey provides a nation-wide infrastructure for efficient sharing of electronic health records. Detailed information on NHIS is given in the following sections. Since NHIS is implemented based on HL7 standards and profiles, first, these standards and specifications are explained.

2.3.1 Health Level Seven (HL7)

Health Level Seven (HL7) [6] is a not-for-profit ANSI [7] accredited Standards Developing Organization. The main purpose of HL7 is to provide standards for the exchange of clinical and administrative data between healthcare systems. HL7 provides standards for interoperability that improve care delivery, optimize workflow, reduce ambiguity and enhance knowledge transfer among all stakeholders, including healthcare providers, government agencies, the vendor community, fellow SDOs and patients.

“Level Seven” refers to the highest level of the International Organization for Standardization (ISO) [19] communications model for Open Systems Interconnection (OSI) [44]; the application level. At the application level of the reference level, in a logical sense, the data exchange between the applications, the timing of the interchange and the communication of certain errors due to the data interchange are addressed.

HL7 defines message structures and trigger events to enable the exchange of the messages. A trigger event causes the transfer of messages between the application systems, that is, when an event occurs in an HL7 compliant system, an HL7 message is sent to other HL7 compliant systems including the necessary data required by the receipants.

Although HL7 Version 2.x is the most widely implemented healthcare informatics standard in the world, it has several interoperability problems. These problems led the HL7 organization to develop a more definitive version almost from scratch, namely the HL7 Version 3.

Version 3 addresses the issues arising from the optional fields by using a well-defined methodology [45] based on a reference information model. Using rigorous analytic and message building techniques and incorporating more trigger events and message formats with very little optionality, HL7's primary goal for Version 3 is to offer a standard that is definite and testable, and provide the ability to certify vendors' conformance. Version 3 uses an object-oriented development methodology and a Reference Information Model (RIM) [46] to create messages. The RIM is an essential part of the HL7 Version 3 development methodology, as it provides an explicit representation of the semantic and lexical connections that exist between the information carried in the fields of HL7 messages.

Reference Information Model (RIM)

An information model is a structured specification of the information within a specific domain of interest. It expresses the classes of information required and the properties of those classes, including attributes, relationships, constraints, and states.

The Reference Information Model (RIM) [46] is the cornerstone of the HL7 Version 3 development process. An object model created as part of the Version 3 methodology, the RIM is a large pictorial representation of the clinical data and identifies the life cycle of events that a message or groups of related messages will carry. It is a shared model between all the domains and as such is the model from which all domains create their messages.

The RIM is comprised of six "back-bone" classes as shown in Figure 2.4. Every happening documented in the healthcare domain is represented by the Act class. Physical things and beings that take part in healthcare are represented by the Entity class. The Role class establishes the roles that entities play as they participate in healthcare acts. The Participation class defines the context for an Act by defining the relationship between Act and Role classes. The ActRelationship class defines the relationship between two instances of the Act class. Similarly, the RoleLink defines the relationship between two instances of the Role class.

The Act, Entity and Role classes are further specialized to subclasses. In the HL7 representation, a new subclass is added to the RIM only when new attributes or associations are needed which are not available in the super classes.

A specialized concept which needs no further attributes or associations is represented by assigning a unique code in the controlling vocabulary to specific attributes. Therefore, these

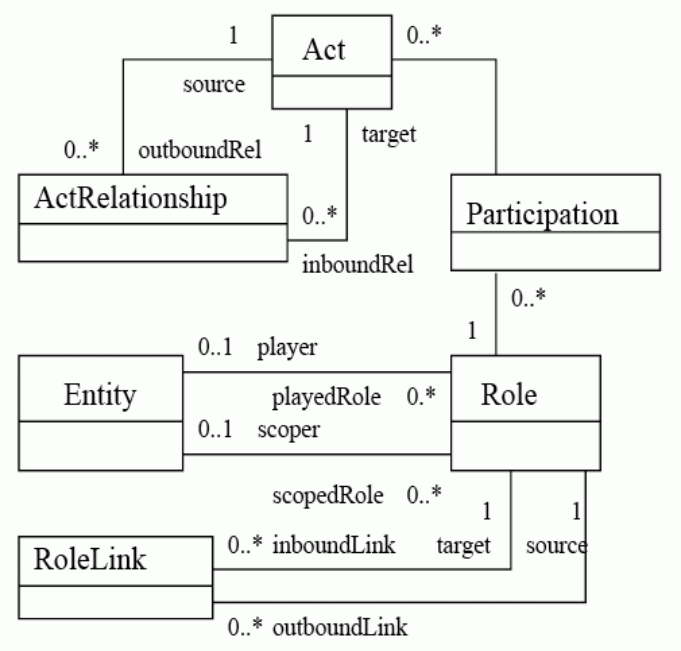

Figure 2.4: RIM back-bone classes

three classes include the following coded attributes, which serve to further define the concept being modeled:

- *classCode* (in Act, Entity and Role) represents the exact class or concept intended, whether or not that class is represented as a class in the RIM hierarchy.
- *moodCode* (in Act) further delineates the Act instance as an occurrence, intent, goal, etc.
- *determinerCode* (in Entity) distinguishes whether the class represents an instance or a kind of Entity.
- *code* (in Act, Entity and Role) provides for further classification within a particular classCode value, such as a particular type of observation within the Observation class. It should be noted that code should be consistent with the classCode.

The other three RIM back-bone classes - Participation, ActRelationship and RoleLink - are not represented by generalization-specialization hierarchies. Nevertheless, these classes represent a variety of concepts, such as different forms of participation or different kinds of

relationships between acts. These distinctions are represented by a *typeCode* attribute that is asserted for each of these classes. For example, the "author" concept which describes the party that originates the Act can be derived by assigning the "AUT" to the Participation class of the RIM.

An example application of these RIM classes with the rough adaptation of Pregnant Observation Minimum Health Data Set from the National Health Data Dictionary of Turkey [47] is represented in Figure 2.5. In this example, the main Act class is the "Pregnant Observation" which has three participations as the subject, performer and observer. "Pregnant Observation" may be the cause for a "Procedure". Moreover, there is a "direct authority over" relationship between the performer and observer doctors.

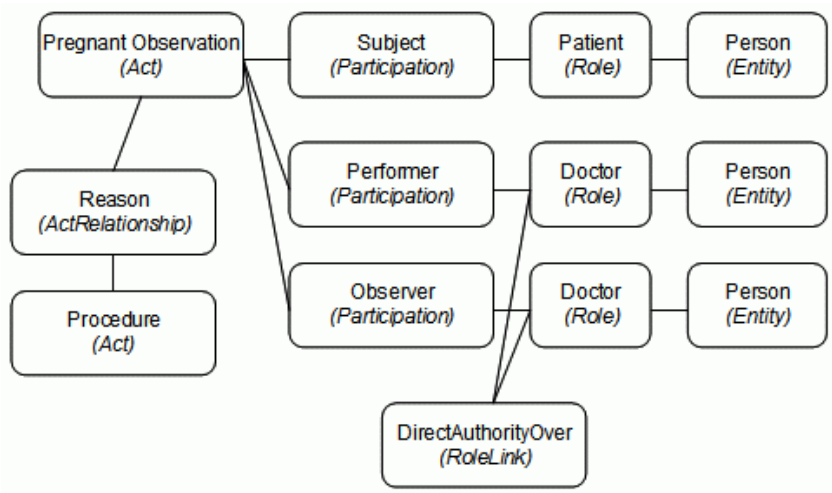

Figure 2.5: RIM classes in action

Message Development Framework (MDF)

HL7 Message Development Framework (MDF) [45] provides a methodology for developing HL7 messages for HL7 Version 3.x. It is a reference manual that describes each step of message construction, how to use the tools that support this process, and the concepts involved. MDF is used by members of HL7 Working Group.

In HL7 Version 3, RIM is the source of all message contents. Figure 2.6 shows how message structures are defined based on the RIM and the MDF.

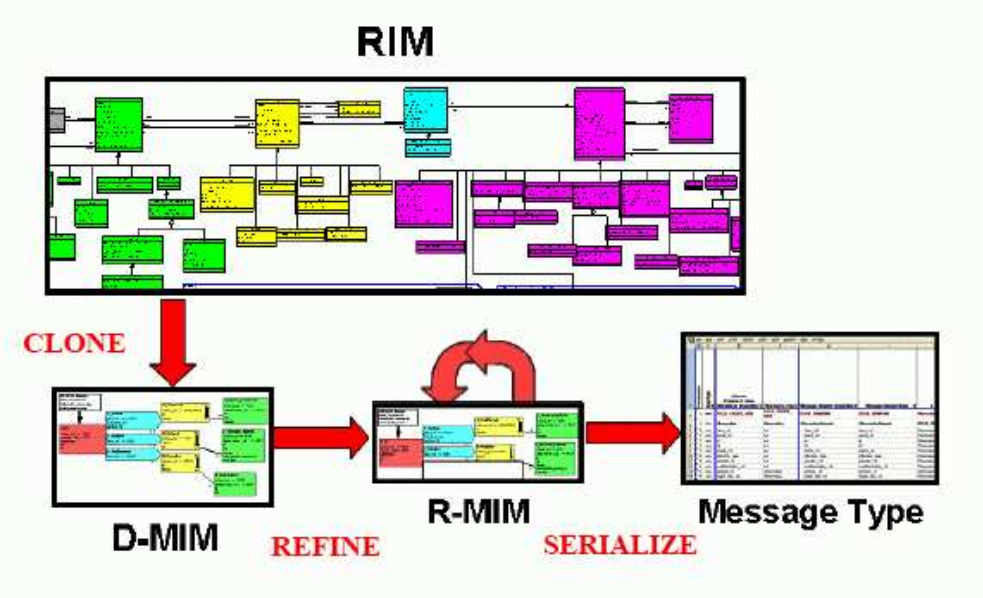

Figure 2.6: Message Structure Generation in HL7 Version 3

HL7 also provides an XML Implementable Technology Specification [48] to express HMD in XML Schema Definitions (XSD) [13]. HL7 defines several message structures in various domains including account and billing, blood bank, clinical genomics, claims and reimbursement, laboratory etc. In addition, HL7 also specifies Clinical Document Architecture which describes the structure and semantics of clinical documents exchanged between healthcare providers.

Refinement, Constraint and Localization

The HL7 methodology uses the Reference Information Model (RIM) and the HL7-specified Vocabulary Domains [49], and the Version 3 Data Type Specification [50] as its starting point. It then establishes the rules for refining these base standards to arrive at the information structures that specify Message Types and equivalent structures in Version 3.

The Refinement, Constraint and Localization [51] specification addresses:

- the “rules” and processes for refining the standard as described in the Message Development Framework (the process leading from RIM to HMD and XSD respectively) through constraint and extension, including which standard artifacts are subject to constraint or extension

- the definition of constraint and localization profiles
- the criteria for establishing a conformance statement

A profile is a set of information which is used to document system requirements or capabilities from an information exchange perspective. The documentation is expressed in terms of constraints, extensions, or other alterations to a referenced standard or base profile. The categories of profiles in HL7 include annotation, constraint, implementable, conformance, localization and conflicting profiles. All of these profiles require the documentation of the formal constraints, extensions and annotations that are applied through the message development process.

To guarantee interoperability, only the authorized HL7 Technical Committees can start the refinement process from the RIM and apply the mentioned constraints while the implementers of HL7 are recommended to start from the HMD although R-MIM is allowed as well.

Transport Specifications

Until Version 3, HL7 did never deal with OSI layers under the application layer; however starting with Version 3, HL7 has shown interest in the transport layer of the OSI layers. The HL7 Message Transport Specifications [52] provide details as to the usage of a variety of communication transports for the exchange of HL7 based content, messages and documents.

Currently HL7 v3 recommends three transport mechanisms to exchange HL7 messages:

1. ebXML Messaging Profile [53]
2. Web Services Profile [54]
3. TCP/IP based Minimum Lower Layer Profile (MLLP) [55]

These Transport Specifications are not to be confused with the content of Transmission Infrastructure. Transmission infrastructure describes the information model, messages and interactions related to the assembly of an HL7 v3 composite message. The Transport Specifications address moving the message payload (the HL7 v3 composite message and/or HL7 v2 composite message) from sender to receiver. These transports are all capable of moving HL7 v3 composite messages and may also support moving HL7 v2 and CDA composite messages.

In NHIS, HL7 Web Services Profile is used for the transportation. Therefore, this thesis is strongly related with the HL7 Web Services Profile.

HL7 Web Services Profile

Web Services are a way for applications to expose software services using standard interoperable protocols, regardless of the platform on which they are implemented. Advanced Web Services Protocols (WS-* Protocols) are built on top of the foundation for Web Services constituted by XML, SOAP and WSDL [56] to express additional functionalities. These are specifications that are developed with the intention of broad adoption and interoperability and focus on security, reliability, transactions, description, discovery and other capabilities.

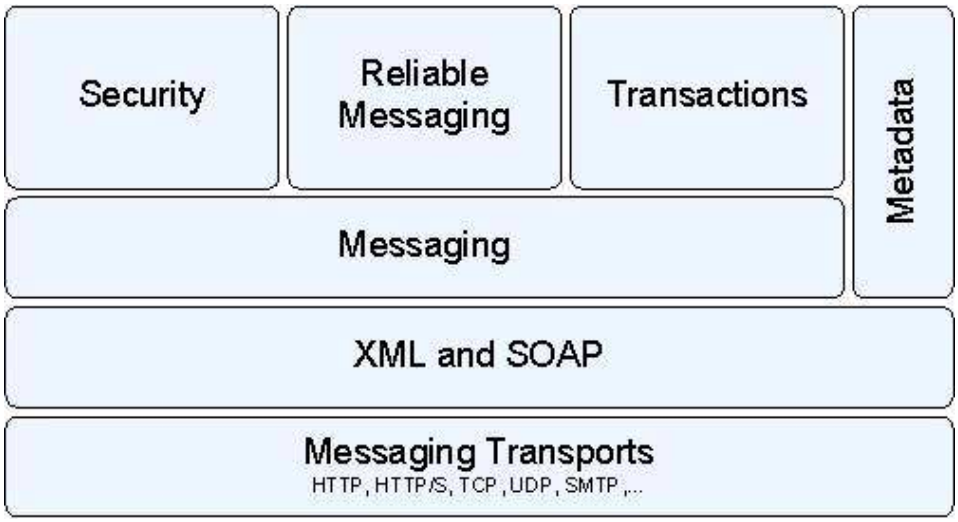

Figure 2.7: Relation between WS-* standards [54]

Figure 2.7 shows the architecture and the relation between the different WS-* protocols [57]. At the bottom of the stack, different network transports provide connectivity between applications and service consumers and providers. The rest of the WS-* specifications are largely independent from the specific network transport chosen.

The profile sets some rules about the invocation of the web services deployed by the profile roles, the usage patterns of the WS-* standards on these deployments and other specifications to support an interoperable communication. One of the rules which is set by the profile is as follows:

"HL7 documents are transported in the SOAP Body, under a single element, which is the top-level element of the original HL7 XML message. Both SOAP 1.1 and SOAP 1.2 [41] are allowed as format for the Envelope. The top-level element of the HL7 message must be embedded as the only child of the soap:Body element."

One of the most important aspects of the Web Services Profile is the use of WS-Security related patterns over the messages. This profile says that ''Unless otherwise noted, the guidance provided in this profile, and the individual WS-* security specifications must be followed in addition to the HL7-specific guidance" . However, this profile does not provide complete guidance on every aspect of Web services security. For example, it does not address how to find and use a certificate authority, manage X.509 certificates, manage trust relationships, deal with delegation scenarios, or use transport-level security.

The first requirement of the profile dictates the authentication and authorization processes. According to the profile all parties must be authenticated in a secure message exchange. WS-Security set provides a number of mechanisms for authenticating and authorizing the message senders based on the security tokens attached to each message. There are a variety of different security token types including Username Tokens, binary security tokens (e.g., X.509 certificates, Kerberos tickets, etc.), XML-based security tokens (e.g., SAML, REL, etc.), and generated security context tokens. All of these token types may be used for authentication and authorization purposes. Messages that do not contain or reference at least one security token cannot be authenticated, authorized, or secured with signatures and encryption. [54]

Since NHIS services adopt the HL7 Web Services profile, it is a must for the FMISs and HISs to use the WS-UsernameToken Profile, which is the selected authentication and authorization framework within NHIS. The testing approach of this thesis implements mechanisms to test the convenient usage of WS-UsernameToke constructs inside the "Transmission Schema" instances.

2.3.2 HL7 Clinical Document Architecture (CDA)

HL7 Clinical Document Architecture (CDA), previously called Patient Record Architecture (PRA), is a document markup standard that specifies the structure and semantics of a clinical document (such as a discharge summary or progress note) for the purpose of exchange

[8]. A clinical document includes clinical observations and services about care events. A valid CDA document is encoded in Extensible Markup Language (XML) and conforms to the CDA Schema which is derived from the CDA Hierarchical Description based on the XML Implementable Technology Specification. The CDA Hierarchical Description is derived from the CDA R-MIM through the process shown in Figure 2.6. In other words, HL7 RIM is the source of the structure and semantics of a CDA document.

So far, HL7 has released two versions of CDA. The CDA Release One (CDA R1) is the first specification derived from the HL7 RIM. It became an ANSI approved HL7 Standard in 2000. The CDA Release Two (CDA R2) became an ANSI-approved HL7 Standard in 2005 [58]. In this thesis, CDA R2 is referred whenever the term CDA is used.

A CDA document has two main parts, the header and the body. In the CDA R1, only the header part is derived from the RIM. In the CDA R2, in addition to the header part, the clinical content in the document body is also derived from the RIM. Therefore the CDA R2 model enables the formal representation of clinical statements through CDA Entry classes.

A CDA header defines the context of the document by providing information on authentication, the encounter, the patient, and the involved providers whereas the CDA body includes the clinical report. The body part can be either an unstructured blob or a structured hierarchy which involves one or more section components. Within a section, narrative blocks and CDA entries are defined. Machine-processable clinical statements are represented by these CDA entries whereas the narrative blocks are human readable forms of these clinical statements. Figure 2.8 [8] depicts the major components of a CDA document.

The “ClinicalDocument” is the root element of the document. The header is located between the <ClinicalDocument> and the <structuredBody> tags. There exist sections in the ”structuredBody” element. Each section can contain a single narrative block located in the “text” element. A narrative block is the human readable portion of the section when rendered with an appropriate stylesheet. Sections also contain CDA entries which are used to represent structured content. CDA entries are machine-processable portions of the sections.

```
<ClinicalDocument>
  ... CDA Header ...
  <structuredBody>
    <section>
      <text>...</text>
      <observation>...</observation>
      <substanceAdministration>
        <supply>...</supply>
      </substanceAdministration>
      <observation>
        <externalObservation>...
        </externalObservation>
      </observation>
    </section>
    <section>
        <section>...</section>
    </section>
  </structuredBody>
</ClinicalDocument>
```

Figure 2.8: Major components of a CDA document

2.4 National Health Information System of Turkey (NHIS)

Turkey's National Health Information System (NHIS) Project [4] was initiated on January 30, 2003 with the participation of representatives from governmental institutions, non-governmental organizations, universities and the private sector under the coordination of Ministry of Health in order to establish cooperation among the sectors and national health information system's infrastructure.

NHIS is based on sharing a functional database which is accessible by authorized people and institutions with defined access rights that covers all the citizens' records from the birth and throughout his/her life on a spine of communication network with high bandwidth throughout the entire country and using the technologies reaching telemedicine applications in professional practice.

Based on the pre-defined goals of NHIS, several achievements have been done which are summarized in the following sections. More information is available in our publications [59] and [58].

Sağlık-Net: The National Health Network

Sağlık-Net is the conversion of the existing LAN-WAN into a true health network platform providing linkages, services and data repositories (e.g. minimum data sets of Electronic

Healthcare Records) to all authorized parties in the health sector. This National Health Platform should be recognized, respected and trusted as the secure national platform for everything that is Health Information, either systems or services or both.

Sağlık-Net is operational now and it is still being developed. The NHIS is built upon Sağlık-Net and as of May 2009, the progress is as presented in Figure 2.9.

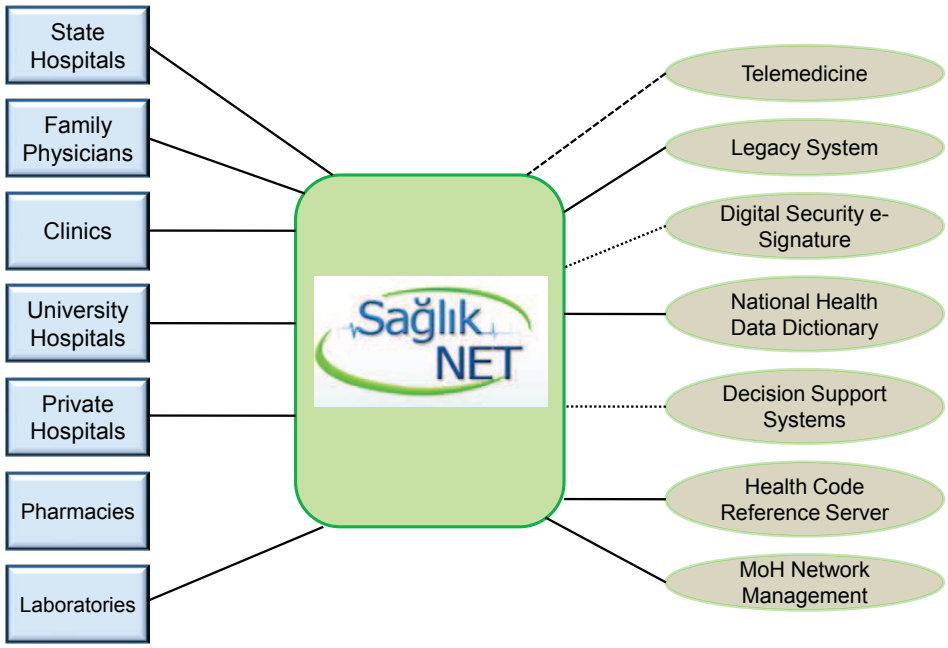

Figure 2.9: Sağlık-Net as of May 2009

As shown in Figure 2.9, the National Health Data Dictionary (Ulusal Sağlık Veri Sözlüğü, USVS) [47], the Health Coding Reference Server (Sağlık Kodlama Referans Sunucusu, SKRS) [60], legacy systems (e.g. Personnel, Financing, Health Statistics) and some network management components are already connected to the Sağlık-Net. Telemedicine applications and the more comprehensive digital security mechanisms on top of the current Web Services Security module are under development to be connected.

On November 17, 2008, the organizations that are ready to send information to the servers located at the Ministry of Health premises have started to send the HL7 CDA messages. The software companies, developing Family Medicine Information Systems (FMIS) and Hospital Information Systems (HIS) in Turkey have to comply with the standards developed by the Ministry of Health. In this way, interoperability among NHIS servers and various Hos-

pital/Laboratory/Clinic/etc. information systems are provided. The Electronic Healthcare Records are based on HL7 Clinical Document Architecture (CDA) and use the National Health Data Dictionary, and the relevant coding systems. As of January 15, 2009, that is the official date published by the Ministry of Health of Turkey, the care organizations have started to send the real data to the servers. Currently, nearly all of the state hospitals of Turkey are successfully sending the daily messages to the NHIS servers. University hospitals and primary care institutions are continuously boosting up their integration capabilities with Sağlık-Net. Electronic Health Records of Turkish citizens are now being populated on the NHIS servers. As statistical information, on March, 2009, on the average 100.000 - 150.000 messages arrived to the NHIS servers, on the daily basis, from all the connected institutions. 80 percent of this incoming data is formed of examination data set.

In the following sections, the major components of the Sağlık-Net, some of which are above-mentioned, are described.

The Health Coding Reference Server (HCRS)

In order to provide common coding/classification systems that are available to all healthcare players, MoH Department of Information Processing developed the Health Coding Reference Server (HCRS) [60] which encapsulates all the international and national coding systems used in Turkey within a publicly accessible server.

Some of the coding systems available from HCRS are ICD-10 [61], Drugs, ATC (Anatomic, Therapeutic, and Chemical Classification System), Associations, Clinics, Specializations, Careers, Health Application Instructions, Supplies, Vaccines, Baby Monitoring Calendar, Pregnant Monitoring Calendar, Child Monitoring Schedule and Parameters.

The concept of Health Coding Reference Server (HCRS) is similar to the “vocabulary domain / value set” mechanism of HL7 v3. All the software companies doing business for Turkish health market are obliged to use the HCRS in their software and to design their products for fast adaptation to the latest updates in the HCRS, latest in 7 days from the date of update.

The National Health Data Dictionary (NHDD)

The National Health Data Dictionary (NHDD) [47] is developed to enable the parties to share the same meaning of data, and use them for the same purpose. The data whose definition

and format determined within the NHDD establishes a reference for the information systems used at health institutions. Thus, the content interoperability among different applications is provided through the NHDD.

NHDD is composed of data sets and data elements conforming to ISO/IEC 11179-4 Standard [62]. Currently, there are 46 Minimum Health Data Sets and 261 data elements.

The data groups used for data collection are called Minimum Health Data Set (MHDS) and are formed from the NHDD as shown in Figure 2.10. In other words, MHDSs define the data sets which emerge at the time of presenting a certain service, for example, Infant Monitoring Data Set or Pregnant Monitoring Data Set. Some example MHDS are “Citizen/Foreigner Registration MHDS”, “Medical Examination MHDS”, “Prescription MHDS”, “Pregnant Monitoring MHDS”, “Cancer MHDS” and “Inpatient MHDS”.

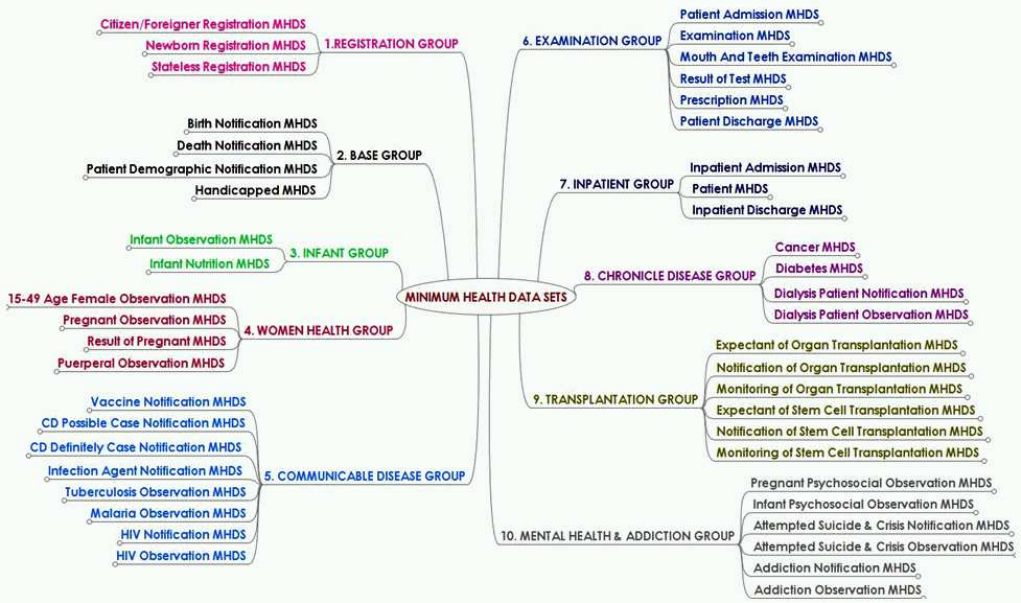

Figure 2.10: Minimum Health Data Sets in the NHDD

The data elements within the Minimum Health Data Sets are mostly coded with coding systems and all these coding systems are available at the Health Coding Reference Server (HCRS). If a data element is defined in the National Health Data Dictionary as coded or classified, then the related coding/classification system is given both within the definition of the data element and in the “HCRS System Code” field. There are two possibilities for a coded element: either the value is gathered from a coding system such as ICD-10, healthcare insti-

tutions, specialties, etc. or the value is of parametric kind such as gender, or marital status.

The Healthcare Professional Registry

Ministry of Health is authorized to provide the work licenses to the physicians in Turkey. The diploma/specialty information of the medical professionals is recorded together with their Turkish citizenship numbers in the Doctor Data Bank (DDB) [63].

The Doctor Data Bank, that is, the Healthcare Professional Registry serves two purposes: The first one is that most of the payment providers control the health service and the prescriptions according to the physicians' specialty. For example, when a rule indicates that only the physicians with a certain specialty can prescribe certain medicines, it becomes possible to check whether the doctor who has signed the prescription has the required specialty.

Data Collection and Sharing in the National Health Information System

The National Health Data Dictionary, the Minimum Health Data Sets and the Health Coding Reference Server provide the information space used in the messages to be exchanged between the peripheral systems and National Health Information System. It should be noted that these data sets do not have a wire format and the most important decision to be taken at such nation-wide projects is whether to use a standard format. Most of the time, standards do not cover all of the identified information requirements in a project; therefore, the implementers choose to develop their own proprietary format instead of a standard format. However, this decreases interoperability. In the National Health Information System, HL7 v3 is selected because of the following reasons:

- HL7 is the most widely used electronic healthcare standard. Although, National Health Information System is to be used locally in Turkey, when it comes to communicate with other countries, the systems should be ready.
- After the completion of the NHIS, all of the medical information systems used in the nation-wide healthcare institutes should be adapted to communicate with the NHIS. Being based on a widely-used standard will facilitate the interoperability to a large extent.
- HL7 v3 provides mechanisms to extend the messages according to the requirements of a project.

- Specifically, the version 3 of the HL7 standard is selected rather than HL7 v2.x because of optionality problems of v2.x. Additionally, HL7 v3 is a standard whose conformance can be tested. In other words, it becomes possible to test the software clients running on the peripheral medical institutes that provide data to the NHIS servers. This is not possible in the 2.x versions of HL7.

Development of the Transmission Schemas

In the current version of the NHIS, the Transmission Schema instances are regarded as HL7 v3 messages and localized according to the Turkey's HL7 Profile [5].

A "Transmission Schema" contains a main Minimum Health Data Set (from which the transmission schema is named after) and a set of auxiliary MHDSs that helps the interpretation of the main MHDS. An example transmission schema for "Examination" defined in the National Health Data Dictionary is shown in Figure 2.11 where the "Examination" data set is sent together with the "Newborn Registration" or "Citizen/Foreigner Registration" data sets. Furthermore, "Patient Admission" and "Patient Discharge" data sets are also required. If there are any "Test Result" data sets or "Prescription" data sets, they are also sent along with the "Examination" data set.

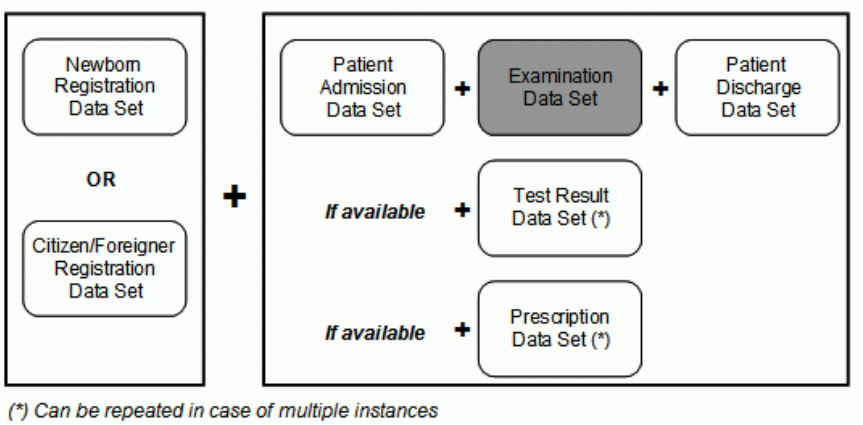

Figure 2.11: The Examination Transmission Schema from the NHDD

It should be noted that there is no specific HL7 v3 Domain for all of the Transmission Schemas. For example, there is no HL7 Domain that is suitable for "Pregnant Psychosocial Observation" or "Communicable Disease Probable Case Notification". Therefore, the CDA, which provides a generic mechanism to identify the contents of electronic healthcare

documents, is selected as the document format.

```
<examination classCode="DOCCLIN" moodCode="EVN">
   <id root="2.16.840.1.113883.3.129.2.1.3" extension="11333439-08ab-42c5-ec2d-17064c153456" />
   <code code="MUAYENE" codeSystem="2.16.840.1.113883.3.129.2.2.1" codeSystemName="Döküman Tipi"
codeSystemVersion="1.0" displayName="Muayene MSVS (Vatandaş/Yabancı)" />
```

Figure 2.12: The Beginning of the Examination Transmission Schema

Each “Transmission Schema” is wrapped with a root element named after the main data set in the transmission. For example, as shown in Figure 2.12, the root tag of the “Examination Transmission Schema” is <examination>. In this example, the document type (“Doküman Tipi” in Turkish) is “Examination” (“MUAYENE” in Turkish) and this code is obtained from the DocumentType-CS Code System whose object identifier (OID) is “2.16.840.1.113883.3.-129.2.2.1”.

```
<examinationDataset classCode="DOCSECT" moodCode="EVN">
   <id root="2.16.840.1.113883.3.129.2.1.4" extension="4e7e0004-8e9e-44d2-9a9e-099d071e646a" />
   <code code="MUAYENE" codeSystem="2.16.840.1.113883.3.129.2.2.2" codeSystemName="Veriseti"
codeSystemVersion="1.0" displayName="Muayene Veriseti" />
```

Figure 2.13: An Example First-level Section for the Examination TS

The “Data Sets” in the “Transmission Schemas” correspond to the first-level “Sections” in the CDA. The name of the opening tag of the “Data Set” is obtained by concatenating the name of the dataset with the “Dataset” keyword. The “code” of a “Data Set” is retrieved from the DataSet-CS Code System. For instance, in the example given in Figure 2.13, the opening tag is <examinationDataset> and the code is specified as “Examination” (“MUAYENE”). This code is obtained from the DataSet-CS (“Veriseti”) Code System whose object identifier (OID) is “2.16.840.1.113883.3.129.2.2.2”.

The data sections that wrap the NHDD data elements are represented by nesting new “section” elements in the data set’s “section” elements. The opening tag of a data section is obtained by concatenating the data element’s name with the “section” keyword. For example, the diagnosis (“TANI” in Turkish) data element is introduced to the examination data set with the <diagnosisSection> XML element as shown in Figure 2.14.

```
<examinationDataset classCode="DOCSECT" moodCode="EVN">
   <id root="2.16.840.1.113883.3.129.2.1.4" extension="4e7e0004-8e9e-44d2-9a9e-099d071e646a" />
   <code code="MUAYENE" codeSystem="2.16.840.1.113883.3.129.2.2.2" codeSystemName="Veriseti"
    codeSystemVersion="1.0" displayName="Muayene Veriseti" />
   ...
   <diagnosisSection classCode="DOCSECT" moodCode="EVN">
      <id root="2.16.840.1.113883.3.129.2.1.5"
       extension="6c4b1e87-c4f4-42b0-aaeb-05e23a77ed5a" />
      <code code="TANI" codeSystem="2.16.840.1.113883.3.129.2.2.3" codeSystemName="Veri Kismi"
       codeSystemVersion="1.0" displayName="Tani Versinin Oldugu Bolum" />
      <text>Hastada vaskuler bagirsak bozuklugu teshis edilmistir.</text>
      <component typeCode="COMP" contextConductionInd="true">
         <diagnosis moodCode="EVN" classCode="OBS">
            ...
            <value code="K55" codeSystem="2.16.840.1.113883.6.3" codeSystemName="ICD-10"
             codeSystemVersion="1.0" displayName="BARSAGIN VASKULER BOZUKLUKLARI" />
         </diagnosis>
      </component>
   </diagnosisSection>
...
```

Figure 2.14: An Example Second-level Section for the Examination TS

As shown in Figure 2.14, the values for the data elements are given with CDA Entry classes such as Observation or Procedure and they are associated to the related “section” element through the “component” element.

Figure 2.15 summarizes the relationships between the artifacts of NHDD, the “Transmission Schemas” and the HL7 CDA R2. Once this mapping is defined, the constraints implied through these mappings are reflected to the schemas by modifying the CDA Level One schema.

This mapping also briefly summarizes the changes that are applied to the original HL7 CDA R2 schema.

Development of the Communication Infrastructure

A “Transmission Schema” instance constitutes a message’s payload. In other words, “Transmission Schema” instances should be encapsulated in messages. For this purpose, “HL7 Transmission and Control Act Wrapper” is used. Example Transmission Wrapper and Control Act Wrapper can be found in Figure 2.16.

HL7 Version 3 provides three transport specifications - ebXML, Web Services and MLLP - for the exchange of HL7 based content, messages and documents. Among them, Web Services Profile is the most promising, as it is based on widely-used Web Services Technology.

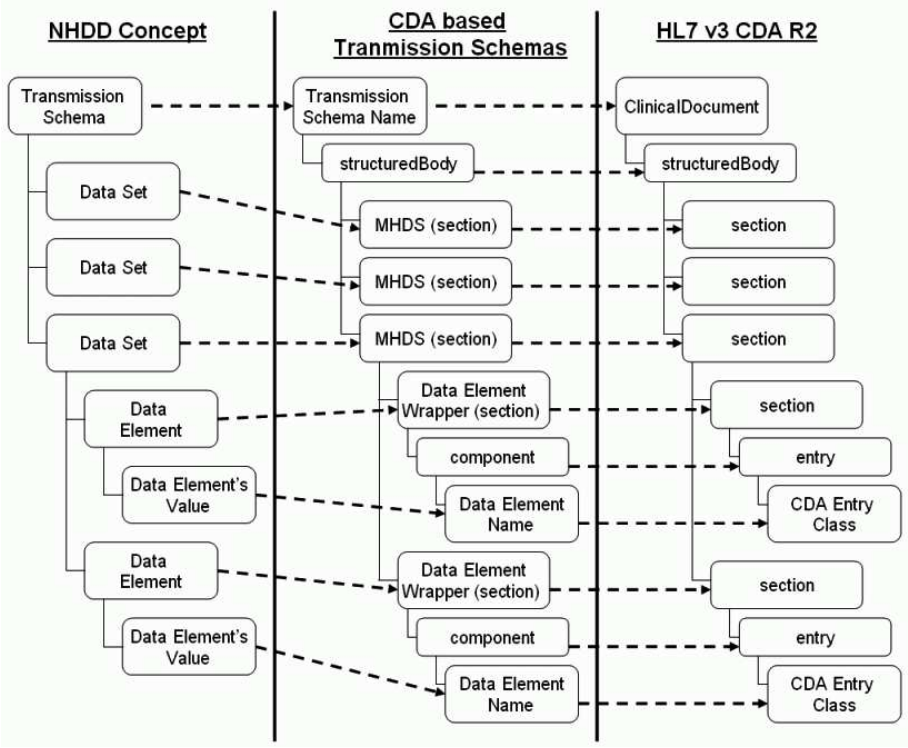

Figure 2.15: Mapping NHDD Concepts to HL7 v3 CDA R2

```
<MCCI_IN000001TR01 xmlns="urn:hl7-org:v3">
  <id root="2.16.840.1.113883.3.129.2.1.2" extension="5b50d046-b567-4fba-a365-60c220c9fae1" />
  <creationTime value="20080402102643" />
  <responseModeCode code="Q" />
  <interactionId root="2.16.840.1.113883.3.129.2.1.1" extension="MCCI_IN000001TR01" />
  <processingCode code="P" />
  <processingModeCode code="T" />
  <acceptAckCode code="AL" />
  <receiver typeCode="RCV">
    <device classCode="DEV" determinerCode="INSTANCE">
      <id root="2.16.840.1.113883.3.129.1.1.5" extension="USBS" />
    </device>
  </receiver>
  <sender typeCode="SND">
    <device classCode="DEV" determinerCode="INSTANCE">
      <id root="2.16.840.1.113883.3.129.1.1.5" extension="SRDC-Deneme" />
    </device>
  </sender>
  <controlActEvent classCode="CACT" moodCode="EVN">
     <subject typeCode="SUBJ">
        <!-- CDA based Transmission Schema instance -->
        <examination classCode="DOCCLIN" moodCode="EVN">
        ...
```

Figure 2.16: Example Transmission Wrapper and Control Act Wrapper from the NHIS

Therefore, in the National Health Information System implementation, Web Services Profile is used for the communication infrastructure. The Basic Profile and the Security Profile of Web Service Profile have been implemented. For security, WS-Security Username Token Profile [64] over Secure Sockets Layer (SSL) is used.

Details of NHIS Web Services

In the current version of the NHIS implementation there are 25 HL7 Web Services. These are: “15-49 Age Female Observation”, “Mouth and Teeth Examination”, “Vaccine Notification”, “Infant Nutrition”, “Infant Observation”, “Infant Psychosocial Observation”, “Communicable Disease Definite Case Notification”, “Communicable Disease Probable Case Notification”, “Diabetes”, “Dialysis Notification”, “Dialysis Observation”, “Birth Notification”, “Pregnant Observation”, “Pregnancy Termination”, “Pregnant Psychosocial Observation”, “Patient Demographics Notification”, “Cancer”, “Puerperal Observation”, “Examination”, “Death Notification”, “Test Result”, “Citizen/Foreigner Registration”, “Stateless Person Registration”, “Newborn Registration” and “Inpatient”.

Almost for each HL7 Web Service there are four operations; namely “Insertion”, “Update”, “Deletion” and “Query” operations. All of these individual operations are synchronous but overall, the NHIS behaves asynchronously. The clients, which are the vendor applications deployed at healthcare organizations’ premises, send their “Transmission Schema” instances through the insertion operation. This operation performs only syntax validation against the related schema and responds with an acknowledgement about the result of the validation. If the invocation of the first operation is successful, then the message is stored at the NHIS servers for detailed processing. This detailed processing involves the semantic validation of the content of the message, which is briefly described in the following “Validation of the Transmission Schema Instances” section. Then, at any time, the client invokes the query operation to query the semantic validation result by using the previously sent document’s Universally Unique Identifier (UUID) [65]. If the detailed content processing of the document is successful, a positive acknowledgement is received from the query operation; otherwise, the errors encountered in the semantic validation phase are reported to the user. The clients are also able to update and delete the previously inserted documents with the help of update and delete operations. The UML Interaction Diagram of NHIS HL7 Web Services in Figure 2.17 shows these interactions and the application roles used in Turkey.

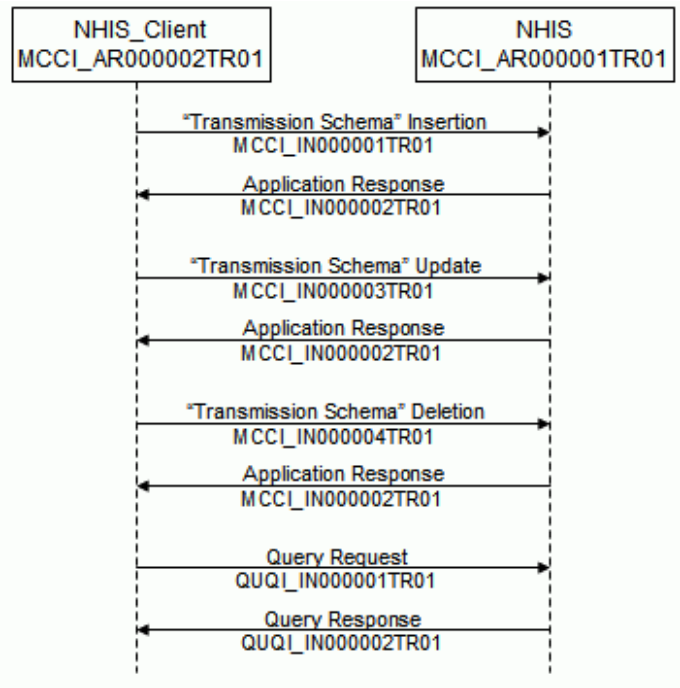

Figure 2.17: Interaction Diagram of NHIS HL7 Web Services

Apart from the HL7 Web Services, there are 16 more Web Services that neither the services nor their content conform to HL7 or any other eHealth standard. These are native synchronous Web Services usually developed for some risky communicable diseases. "Malaria Notification" and "Tuberculosis Notification" are two such examples. They have Web-based forms on the MoH central servers as well and it is expected that healthcare professionals will send these comparably rare observations through these forms. Currently, these 16 non-HL7 Web Services are not in use because they are not deployed on the NHIS servers. These services have faced with significant semantic and syntactic changes. Therefore, new implementations and deployments of these services are in progress. In a near future, these services are being planned to be deployed on the NHIS servers.

Validation of the Transmission Schema Instances

A two phase validation technique is applied for the validation of the incoming messages to the NHIS. In the first phase, which is called syntax validation phase, an incoming document instance is validated against the related XML Schema Definition (XSD) of the "Transmission Schema" by the greeting operations; namely the insertion and update operations. If success-

ful, the message is conveyed to the second phase which is called the semantic validation phase.

The semantic validation phase checks the values in the data elements and the relationships between them. The semantic constraints are categorized into five classes:

1. MERNIS Central Demographics Management System: In Turkey, every citizen has a unique identifier and these identifiers are maintained in a system called MERNIS (Central Demographics Management System) [66]. The patient identifiers in the messages should be validated against this system.

2. Doctor Data Bank (DDB): The identifiers of the healthcare professionals that appear in the messages should be validated against this DDB.

3. Value formats: Values of some data elements should obey some specific formats. As an example, HL7 date can be of the form YYYYMMDD.

4. Coded elements: The coded elements should have values from the Health Coding Reference Server (HCRS).

5. Business rules: There are some rules among the message elements such as the examination end date should be later than the examination begin date. There are more complex clinical business rules as well. These rules are defined and documented [67] in collaboration with the healthcare professionals and the administrative staff of the healthcare organizations.

2.5 TestBATN

TestBATN is a software framework which proposes a design and an execution environment and applies a specific testing approach for dynamic, configurable and automated execution of conformance and interoperability testing against B2B standards, profiles or specifications. TestBATN is an abbreviation for “Testing Business Process, Application, Transport and Network Layers”. That is, it enables an interoperability test platform to test all the layers of the interoperability stack.

The work of this thesis is done under the TestBATN project. This thesis engages an important

role within the TestBATN framework by realizing a Test Execution and Monitoring Environment. TestBATN project is supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) [68]. In this section of the thesis, introductory information about the TestBATN framework is presented. The part of TestBATN which directly corresponds to this thesis is described in detail in Chapter 4, as TestBATN CONTROL AND MONITORING ENVIRONMENT.

This work has also enabled the testing of the National Health Information System (NHIS) of Turkey by providing the Test Execution and Monitoring Environment to the software producers in the health arena of Turkey. Therefore, first, the requirements of the testing process of NHIS are identified and the test scenarios are developed within the Test Description Language of TestBATN. According to the requirements of the developed test scenarios, the Test Control and Monitoring Environment within TestBATN oriented to the expected implementation. In Chapter 3, as NHIS TEST SCENARIOS, the testing approach and the test scenarios are presented in detail.

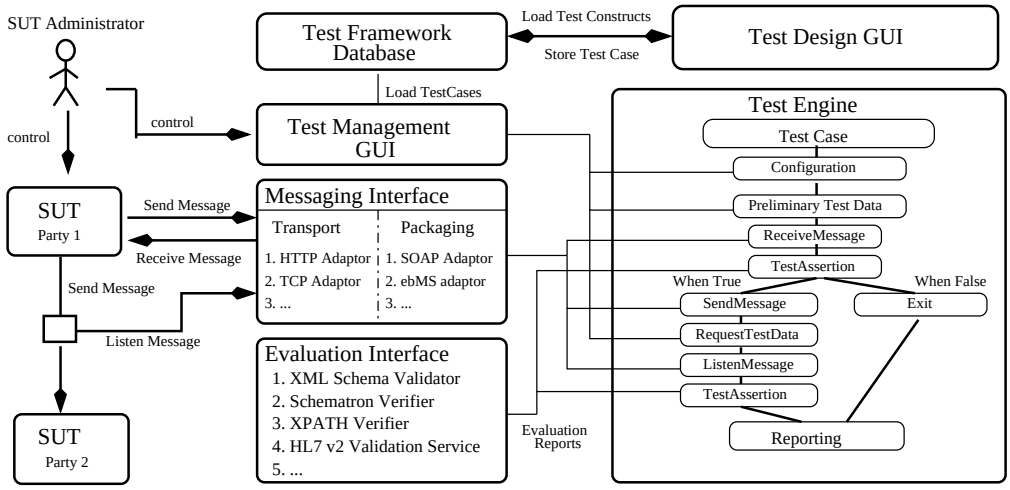

Figure 2.18: Overall Architecture of the TestBATN Framework [69]

Figure 2.18 presents the overall architecture of the TestBATN framework. In the following subsections, some details of the overall picture are described.

TestBATN framework consists of the following features:

- An XML based computer interpretable test language (TestBATN Test Description Language) allowing dynamic set up and automated execution of test cases.

- A Test Execution Model defining the meaning of the behavioral and operational semantics of each Test Description Language (TDL) instruction in an unambiguous way.
- A set of interfaces (TestBATN Module Interfaces) which enables development and integration of pluggable modules supporting different protocols, formats or methodologies and dynamic capabilities to make these modules utilized in execution of the specific TestBATN TDL instructions.
- A Web-based Test Control and Monitoring Environment which enables the users to monitor and drive the execution of test cases. (Chapter 4)
- A Test Design and Framework Management Environment enabling responsible organizations to develop, deploy and maintain test cases, supplementary materials for any B2B (Business-to-Business) standard, specification or profile.

With the above features, TestBATN framework can be used as

i a conformance test platform where vendors can test their systems' conformity remotely over the Web

ii an interoperability test platform where vendors can test the interoperability of their products against each other remotely over the Web, or in a testing event (e.g. ETSI Plugtests [70], IHE Connectathons [71])

iii a means of testing for the certification of products against a B2B standard or profile.

2.5.1 Main Concepts and Testing Approach

TestBATN framework aims at integrated testing covering all layers of interoperability stack; business process layer, document layer and communication layer. Rather than partial testing, that is testing each layer independently with specific tools and integrating the results, TestBATN has an integrated testing approach which tests the restrictions on these three layers of interoperability stack jointly within a test scenario.

Figure 2.19 illustrates the main concepts of the framework and setup for conformance testing. A ***Test Scenario*** is the abstraction for the complete description of the participating entities

and the steps required to achieve a specific test purpose; testing the systems against the conformance or interoperability requirements of a part of a standard or a profile. The TestBATN framework follows the business process terminology and calls the participating systems of the test scenario as parties.

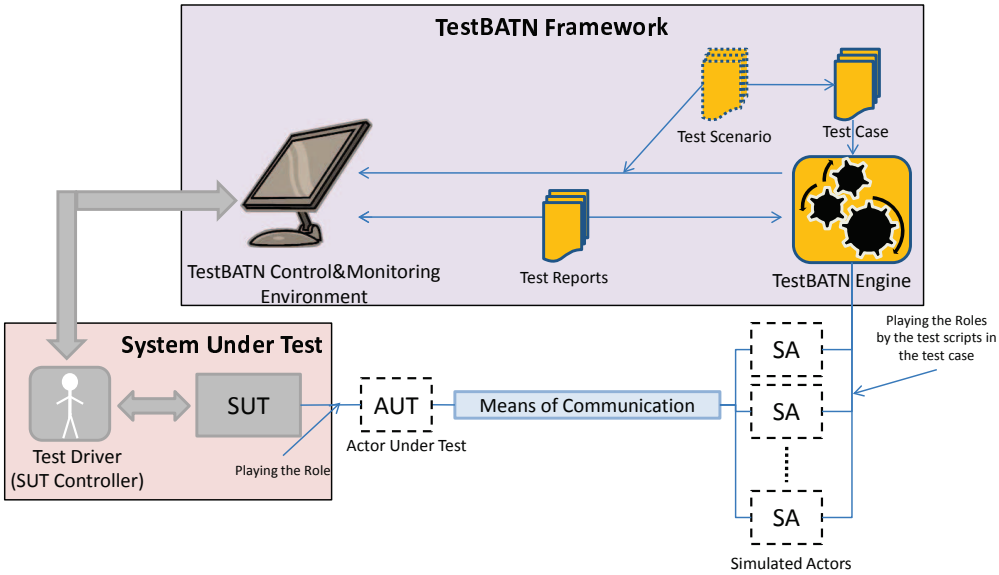

Figure 2.19: TestBATN Conformance Testing Setup

Generally, ***Test Parties*** are the corresponding abstraction of the framework for the actors specified in the business process in the base standard. In the TestBATN framework, Test Parties can be in two types; simulated or Actor Under Test (AUT). The parties, which are specified as AUT, are the entry points for Systems Under Test (SUTs) to participate to the test execution. The simulated parties are the actors that reside in the business process or message choreography specified in base specification, but that are not intended to be tested in the test scenario. They are simulated by the framework by means of test scripts to achieve the test purpose.

The ***Test Steps*** mentioned in the Test Scenario description are the abstraction for actions or instructions that will be performed on the Test Framework or SUT side to execute the test case. The backbone for these steps is the message choreography specified within the Test Scenario (or derived from the base specification conformance or interoperability criteria). Other steps are the instructions for the Human Test Drivers controlling the SUTs and validation steps that

will contribute to the test verdict.

As previously mentioned, Test Scenario is the abstraction and it should be interpreted as the sequence or flow of actions in Test Designer's mind to achieve the test purpose. In order to automate the test execution, this description is defined in TestBATN machine processable language; TestBATN TDL. By following the conformance and interoperability testing literature terminology, this definition which corresponds to a test scenario is called a Test Case.

2.5.2 Conformance Testing

Standardized protocols and services, and the applications developed through the protocols can be formally tested in two related but different ways. One way goes through the conformance testing. Conformance testing depicts whether an application correctly implements a particular standardized protocol or not. In other words, the testing of the conformance shows whether or not a single implementation of the protocol meets the conformance requirements specified for that protocol.

When a software instance includes an implementation of a universally standardized protocol, it becomes possible to specify the test criteria and procedures with a quality comparable to that of the protocol standards themselves. The protocols may set some rules on several layers of the interoperability stack and these rules are formally defined through some formal documentation. XML Schemas may restrict the structure of the content, business profiles may rule on the choreography of the exchanged messages among the parties etc. Therefore, these formal restrictions form a basis for the testing applicability on that of the softwares which implemented the protocols.

Figure 2.19 shows basic details and setup for conformance testing. During the test execution, two entities; TestBATN Engine and TestBATN Execution and Monitoring Environment take role in the framework side. The ***TestBATN Engine*** is the system which interprets and executes the Test Case definition, which is scripted in the Test Description Language, and manages the whole process. The TestBATN Control and Monitoring Environment (Chapter 4) is the interface between the engine and the Human Test Driver to provide real time monitoring on the test execution, some control on the scenario and reporting of the test results. The ***Human Test Driver***, while monitoring the test execution with TestBATN Control and Monitoring

Environment, manages the SUT according to the instructions shown on the Graphical User Interfaces (GUIs).

The purpose of conformance testing is to determine conformance of a single implementation against a particular standard. Therefore, in the setup shown in Figure 2.19, there is only one SUT playing the role of the Actor Under Test (AUT) specified in the conformance test case. The TestBATN Engine on the other hand can simulate several parties according to the business process specified in the base standard or testing purpose of the test scenario. The Means of Communication is the physical setting that enables communication among systems, can be internet, local network or any other instrument.

In any given instance of testing, there will be a test report which includes the final test verdict, intermediate test verdicts for each step, and detailed reports showing the logs and errors for each step. In fact, the reports and the verdicts for each step are shown to the Human Test Driver in run time without waiting the end of the test case. The interface between the TestBATN engine and the TestBATN Control and Monitoring Environment realizes this run-time monitoring facility. In addition, a presentation model is defined in the framework and used to present the test scenario to the Human Test Drivers in a better way. Detailed descriptions on the reports will be discussed in Chapter 3 while concentrating on the application of the developed test scenarios in real-life.

2.5.3 Interoperability Testing

Interoperability is the ability of two or more systems or components to exchange information and to use the information that has been exchanged [1]. More specifically, interoperability is said to exist between two applications when one application can accept data (including data in the form of a service request) from the other and perform the task in an appropriate and satisfactory manner (as judged by the user of the receiving system) without the need for extra operator intervention [2].

The purpose of the interoperability testing is to prove that end-to-end functionality between, at least, two communicating systems is as by required by the standard or a number of standards on which those systems are based. However, one should keep on mind that interoprerability tests are applied at the end-points and on the functional interfaces of the applications which,

that is, interoperability testing can only specify functional behavior. It cannot explicitly cause or test protocol error behavior [3].

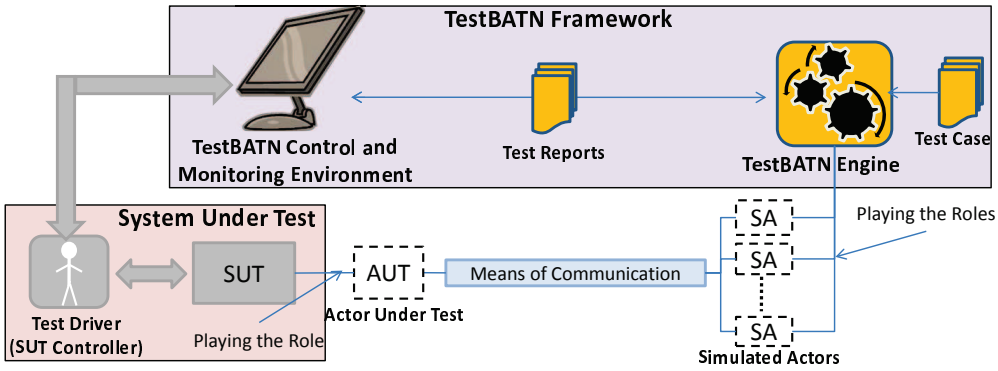

Figure 2.20: TestBATN Interoperability Testing Setup

Figure 2.20 illustrates the interoperability testing setup of the TestBATN when there are two Systems Under Test (SUTs). The situation is similar when there are more than two SUTs. There is not too much difference between the conformance and interoperability testing in the testing process. Since the purpose of interoperability testing is to check the end-to-end functionality between two or more communicating systems, usually there is no simulated actor. However, interoperability test scenarios including simulated actors can be designed for more complex test purposes. Generally, in interoperability test scenarios, the aim is to listen to the messaging between SUTs. The TestBATN framework achieves this by its specific instructions for listening by setting up itself as a proxy. The monitoring and reporting process is the same as in the conformance testing setup.

2.5.4 Scenario Based Testing

It is already mentioned that TestBATN framework approach for both conformance and interoperability testing is integrated testing based on a scenario and the backbone for this scenario is the identified actors and messaging choreography among them. On this scenario skeleton, test designers set further test steps to check conformance or interoperability requirements of the base standard regarding the business documents, other details on business process, and

communication protocols. However, more information on the business collaboration will be needed in order to execute the test scenario. Current approaches provide this information in the narrative test case descriptions before test case executions. Then the test cases are configured in design time accordingly to check these further scenario requirements. TestBATN framework proposes a new approach, scenario based testing which provides mechanisms to set and check these scenario restrictions in the test case definitions.

In the scenario based testing approach, a real life scenario regarding the backbone business process is presented to the Human Test Drivers and they are expected to operate the SUTs according to this scenario. For instance, assume that you need to test a standard which defines the rules for the messaging interaction between healthcare institutions for the sharing of an EHR record for the examination data. The following text can be a part of the scenario requirements:

"*A patient whose name is* **John Doe** *visits doctor Mary in 2008-10-13. The main diagnosis identified after the examination is Liver Cancer ...*"

In this scenario fragment, the underlined words should be the values of some element in the message if this scenario happens in real life. Since the base specification and standard tells where these values should be located in the message (for example with an XSD Schema), for real conformance and interoperability, the application should have the ability to use the information given in real life and generate the message or resulting document correctly with this information. Therefore, in addition to syntactic testing, the semantics of the message should also be tested by scenario based testing.

Scenario based testing is useful but not effective much when all the scenario requirements are provided in design time where the test scripts are configured according to these requirements. In order to make the test scenarios dynamic, configurable and easy to maintain, the TestBATN framework provides a mechanism to present the scenario in structured way like a template. In this way, designation of actual scenario requirements will be delayed until run-time. In TestBATN framework, this phase is integrated in automated test case execution as preliminary phase. When the requirements are specified in test execution time, the test scripts in the test case definition are also configured automatically. The TestBATN framework provides following three mechanisms for the completion of the scenario template:

- An element in the template can be specified in design time by the test designer
- An element in the template can be specified by one of the Human Test Drivers in run-time
- An element in the template can be specified by a special application which can generate randomized values

The Scenario Based Testing approach of the TestBATN framework facilitates test cases which not only guarantee that the application can send some conformant messages, but also guarantee that in any real life scenario the application sends the conformant and semantically correct message. Scenario based testing approach is more important for certification, better interoperability (semantic interoperability) and to ensure accurate data. In other words, by using this approach, it is possible to test if the information in the generated content, that is messages or documents, accurately represents the intentional semantics of the users of the application.

2.5.5 Modular Approach

One of the objectives for TestBATN framework is being a generic testing framework and supporting any B2B standard, specification or protocol for conformance and interoperability tests. Current B2B standards specify a variety of communication protocols, content format or choreographies. In order to support all of those and perform automated conformance or interoperability tests against them, the TestBATN framework is designed as an adaptable and modular software framework. The aim is to leave the dirty job required for testing process to adaptors coded by some programming language. Such adaptors can be developed more easily to perform those jobs. Utilizing specific adaptors to supplement the automated testing process is not new and followed by other test beds or approaches [72]. However, in those approaches, testing adaptors are prepared for specific testing events or specific test cases, and there is no standard methodology to integrate them into the test frameworks.

The TestBATN framework defines interfaces and a standard methodology for the integration of different adaptors into the framework to enhance the reusability of the adaptors among test cases designed for different domains or standards. The approach of TestBATN framework is defining a common standard interface for specific functionalities that need to support differ-

ent protocols, formats, or technologies in testing process or which may facilitate reusability among different test cases from different B2B standards or profiles.

The TestBATN framework defines the following interfaces which will be used in the testing of NHIS.

- **Messaging Interface**
  - Transport/Communication Adaptor Interface: The interface facilitates pluggable adaptors which will be used to receive, send or listen to messages by different protocols like TCP [11], HTTP [20], SMTP [73], etc.
  - Packaging Adaptor Interface: The interface facilitates pluggable adaptors which will be used to pack or unpack messages according to higher layer messaging protocols like SOAP [41], ebMS [74], etc.
- **Test Adaptor Interface**
  - Validation Adaptor Interface: The interface facilitates pluggable adaptors which will be used to validate a content according a schema and which generate a verdict and structured test report for the performed validation. e.g. XML Schema Validator, Schematron Validator
  - Verification Adaptor Interface: The interface facilitates pluggable adaptors which are used to perform complex tests on any content and which generate a verdict and structured test report for the performed tests. e.g. XPATH [15] Verifier, Regular Expression Verifier, other special purpose test tools
- **Value Initiator Adaptor Interface**: The interface facilitates pluggable adaptors which will be used to generate or choose random values from a specific value list. These initiated values are used in run-time message content and scenario requirement generation.
- **Function Library Interface**: The interface facilitates definition and implementation of functions which will be used in TestBATN TDL expressions as auxiliary data processing entities. These functions are similar to XPATH functions used in XPATH expressions.
- **Data Model Interface**: This is the interface to define specific data models for the content formats that are not XML based. TestBATN framework internally transforms

them into Object model and enables expressions and other TDL instruction to run over this data.

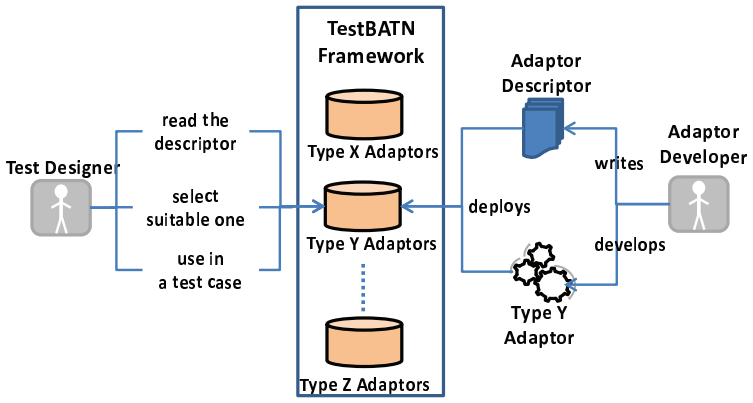

Figure 2.21: Reusability for Adaptors

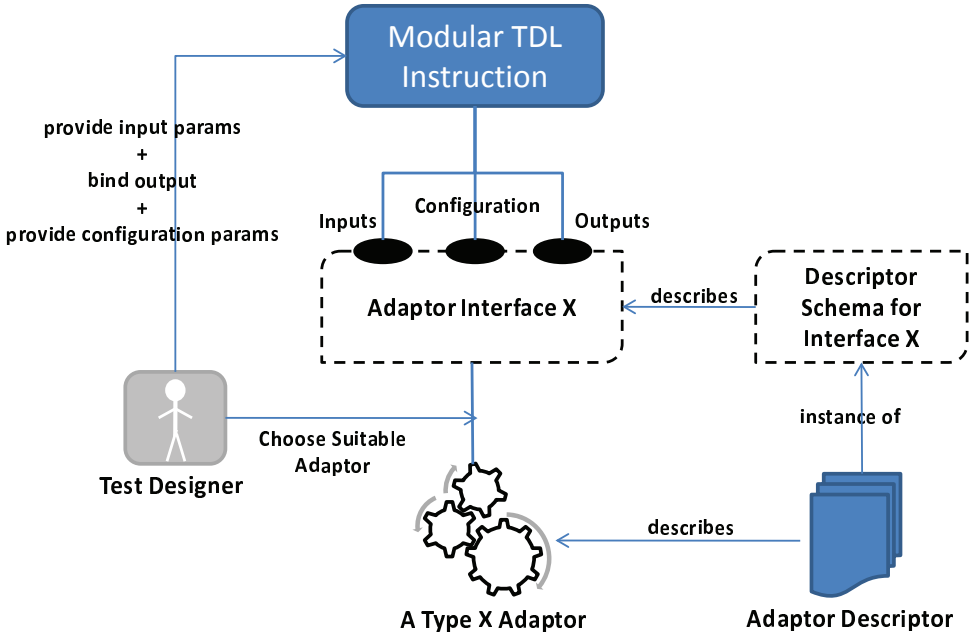

Figure 2.22: Plug-in an Adaptor to a Modular TDL Instruction

Figure 2.21 illustrates the approach of TestBATN framework to enhance reusability for test components or tools. According to the needs for a standard or domain, adaptor developers develop adaptors implementing the corresponding interfaces defined in the framework. They

also write descriptors for their adaptors which describe the main functionality, its input, output and configuration parameters in a structured XML format. When the descriptor and the implementation for the adaptor are deployed into the framework, the adaptor is ready to be used in test case definition process and execution of those test cases.

The Figure 2.22 shows how the Test Designer plugs in a specific adaptor for a TestBATN Test Description Language (TDL) instruction. By considering the test scenario, he chooses the suitable adaptor from the registry of adaptors of that type and mention the ID of the adaptor in the instruction to make the TestBATN engine to use the adaptor while executing the instruction. Then he provides values for input, output and configuration parameters for the adaptor.

CHAPTER 3

NHIS TEST SCENARIOS

Two important inputs to this thesis are introduced in the preceding chapter as National Health Information System (NHIS) of Turkey and the TestBATN Framework. The target outcome of this work is enabling the conformance and interoperability testing of the applications developed within NHIS through TestBATN framework. This chapter focuses on the test scenarios which are developed in the scope of this thesis. The test scenarios are registered to the TestBATN through test suite definitions and published to the Family Medicine Information System (FMIS) and Hospital Information System (HIS) vendors taking part in the health sector of Turkey.

The current version of the NHIS provides 25 HL7 based web services, each of which is specialized to a specific Minimum Health Data Set Transmission as described in [58]. NHIS also published 16 native web services which are not HL7 based. The testing process of this thesis does not deal with those non-HL7 Web services. The services in question are the 25 HL7 Web services.

All of the software systems running in the medical institutes in Turkey, the FMISs and HISs, are obliged to have the ability to transfer EHRs, called “Transmission Schema” instances to the NHIS servers at the Ministry of Health (MoH) premises. In order to guarantee the interoperability, the MoH, published an Implementation/Integration/Interoperability Profile [5] for FMIS and HIS vendors. This profile holds a very important role in the development of the NHIS test scenarios because the standars to be used in the interoperability stack, the code values to be used and several semantic rules are defined through this profile. The profile is based on the following standards and specifications which are described in detail in Chapter 2:

- For transport protocol, HL7 Web Services Profile [54] is used. For security WS-Security Username Token [64] over SSL is required for conformance.

- The “Transmission Schemas” are HL7 CDA R2 [8] conformant EHRs and each HL7 CDA section is a Minimum Health Data Set (MHDS) [75] which is formed from the data elements specified in National Health Data Dictionary (NHDD) [47].

- Health Coding Reference Server (HCRS) [60] serves all coding systems in use in Turkey which is used in the data elements within the Minimum Health Data Sets. For some specific data elements, some other coding systems, like ICD-10 coding system [61], are specified for use.

- For each Transmission Schema and Minimum Health Data Set, several semantic Business Rules are defined to provide consistency among the values used in the data elements.

- In Turkey, every citizen has a unique identifier and these identifiers are maintained in a system called MERNIS (Central Demographics Management System) [66]. The patient id numbers in the messages are required to exist and be consistent with this system.

- In Turkey, every physician is registered to a system called Doctor Data Bank [63]. The id numbers of the doctors in the messages should exist in this system.

The Integration Profile and its reference specifications present all the restrictions and requirements for vendors to update or develop the necessary components within their FMISs and HISs for a successful integration. However, without an extensive and effective testing process this is a difficult job for those vendors. Furthermore, only through testing, correct information exchange among these eHealth applications can be guaranteed and the products can be certified. Conformance and interoperability testing are both important and useful to the testing of the applications whithin NHIS. Conformance testing of the FMISs and HISs can show that those implementations comply with the requirements of the protocols and specifications asserted in the Integration Profile of MoH. However, only testing the conformance of the applications does not guarantee that those implementations will interoperate with each other.

3.1 Testing Requirements

After analyzing the Integration Profile and the eHealth market in Turkey in terms of FMIS and HIS products, the requirements to enable the testing of the systems have been settled [12].

Basic Conformance Testing

The first step in the testing process is to test the ability of the HIS and FMIS systems to send valid "Transmission Schemas" in terms of syntactic and structural constraints. Only after processing a structurally valid message, the applications can go further analysis and processing of the message. Furthermore, since the integration profile sets a number of protocols to be used within each other, the syntactic validation should also be done in a logically consistent order.

The following tests are required to assure the syntactic correctness of the messages:

i Checking the ability of the systems to send HL7 Web Services Profile conformant SOAP (Web Service) messages

ii Checking the ability of the systems to send WS-Security Username Token Profile conformant SOAP (Web Service) message

iii Checking the ability of the systems to use the username and the password assigned to it correctly in the corresponding SOAP header, as specified in the WS-Security Username Token Profile.

iv Syntactic validation of "Transmission Schema" instances sent by the systems against their corresponding XML schemas

v Checking if the code systems and the codes used in the "Transmission Schema" instances are valid. That is, checking whether the code system is one of the code systems specified in the integration profile and whether the code value is valid according to the corresponding code system

Functionality/Semantic Testing

Basic conformance testing guarantees the conformance of the "Transmission Schema" instances sent by a FMIS or HIS to the specified standards and specifications, and hence it

partially guarantees the MoH NHIS servers to accept the transmission and store its content to their database. However, it does not ensure that the information in the transmissions accurately represent the intentional semantics of the FMIS or HIS users (e.g. doctors, family practitioners).

For example, a data element in the “Medical Examination MHDS” is called “Incident Type” which should contain values such as “Normal”, “Emergency”, or “Industrial Accident”. For this data element, the following tests should be performed in order to guarantee the semantically valid transmissions:

i Checking whether the system (FMIS/HIS) is capable of sending different values for the related field. This constitutes to testing whether the system provides all possible code values to its user for selection

ii Checking whether the system (FMIS/HIS) accurately packs the value selected by the user into the transmission

iii Checking whether the system (FMIS/HIS) has the ability to render this value to its users

The testing requirements identified up to this point necessitate scenario based testing where vendors are given with a set of test scenarios and are requested to use their products’ user interfaces to construct a transmission which is conformant with the given scenarios.

Interoperability Testing

Testing the conformance of applications to produce and consume the correct “Transmission Schemas” is necessary but not sufficient in deciding whether a FMIS or a HIS can properly be integrated into Turkey’s NHIS. It is necessary to ensure that the selected options, bindings, and deployment settings of the implementations are compatible across partners [76].

Interoperability testing also enables the testing of the implementation of the NHIS server applications in the terms of the standards and specifications settled in the Integration Profile. However, in this case, testing of the NHIS server applications can only be done through the syntactic checks of the responses generated by the NHIS servers. That is, semantic and functionality checks of the NHIS server applications fall beyond the scope of the interoperability testing in the context of this thesis.

3.2 Testing Approach

The development and application of the test cases adopts a “step-by-step” approach from the basic and simple test cases to the complex ones. According to the National Health Data Dictionary (NHDD) [47] published by the Ministry of Health, Turkey, by means of the Integration Profile, a “Transmission Schema” contains a main Minimum Health Data Set and a set of auxiliary MHDSs that helps the interpretation of the main MHDS. Furthermore, as previously presented in Figure 2.11 which provides an example schematic representation of the Examination Transmission Schema, inside the “Transmission Schema” some of the MHDSs are required while some of them are optional. These are specified through the corresponding XSD Schemas.

Considering the mentioned facts, several test cases are developed through the adopted testing approach of this thesis to realize the testing of FMISs and HISs. Figure 3.1 shows the graphical representation of the testing methodology used in this thesis. The testing tool, which is actually the set of tools within TestBATN, simulates the corresponding NHIS service according to the definition stated inside the developed testcases. The clients configure their applications to send the service messages; those are the “Transmission Schema” instances, to the assigned proxy ports of TestBATN. According to the adopted testing approach, the testcases developed for the use of NHIS clients, start with the basic conformance tests which include the minimum requirements to send a valid “Transmission Schema” message. And the testcases proceed to more complicated semantic and functional scenarios to test different abilities of the FMISs and HISs.

According to a “Tranmission Schema”, one application can send structurally valid messages while meeting the minimum requirements set by the “Transmission Schema”. For example, in the examination “Transmission Schema”, Test Result Datasets and Prescription Datasets are represented as with “if available” tag at Figure 2.11. From this definition, a “Tranmission Schema” instance message including only the required data sets forms a valid message with the minimum requirements. Therefore, basic test cases are developed to verdict on the messages without considering the additional constraints other than the ones specified in the schemas.

The additional constraints can be asserted to test the applications’ abilities on sending mes-

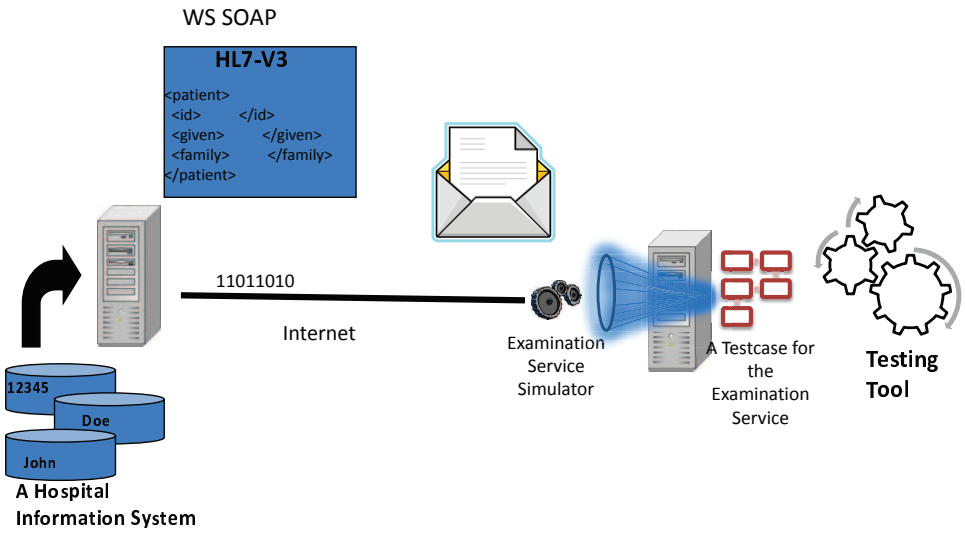

Figure 3.1: Testing Achitecture

sages with non-required, optional MHDSs. For example, occurance of a Prescription Dataset can be enforced through the test case definitions. Afterwards, business and semantic rules can be applied through that optional MHDS inside the “Transmission Schema” instance. This approach constitutes the development of customized basic test scenarios.

Up to now, a FMIS or HIS can be tested through the syntactic and structural checks in addition with the customized basic test scenarios. However, one aspect remains missing. How can we be sure about the interpretation of the data sent within the message? That is, we also need to be sure about the meaning of the data sent at the correct places in the message. An application may send structurally valid messages with meaningless data inside. For example, examination information of the patients from a hospital may be sent to the NHIS servers always with the same diagnosis elements or patient admission elements etc. Therefore, semantically enriched test cases are developed to ensure the semantic correctness of the FMISs and HISs.

As mentioned earlier, NHIS provides 25 HL7 web services for the transfer of EHRs from the healthcare institutions to the MoH servers. All of these services include insert operations which is the main subject to be tested. Apart from the insert operations most of the Web services also include update delete and query operations. Update operation is used to update a previously sent “Transmission Schema” instance and the delete operation is used to remove it

from the main server's databases. The query operation is used to query the processing status of the previously sent message. As mention earlier in section 2.4, NHIS exhibits an asynchronous behavior in total. For each message sent to the 25 HL7 services, a query operation is needed to check whether the message is accepted (if it is valid in syntax and semantics) by the server or not. Furthermore, when a message is accepted by the server, the client applications (FMISs and HISs) have only one chance to query the successful status of the message.

Within the scope of this thesis, testcases for the query, update and delete operations of the available services are also developed.

3.3 Test Case Development

TestBATN provides a well-defined mechanism to register the test case scripts developed with Test Description Language (TDL). Test cases, which specify the test steps through TDL, are each defined in a separate file with an assigned unique identifier in its context and put into a test suite by providing its identifier inside the test suite as shown in Figure 3.2. A single test case definition can only be registered into only one test suite.

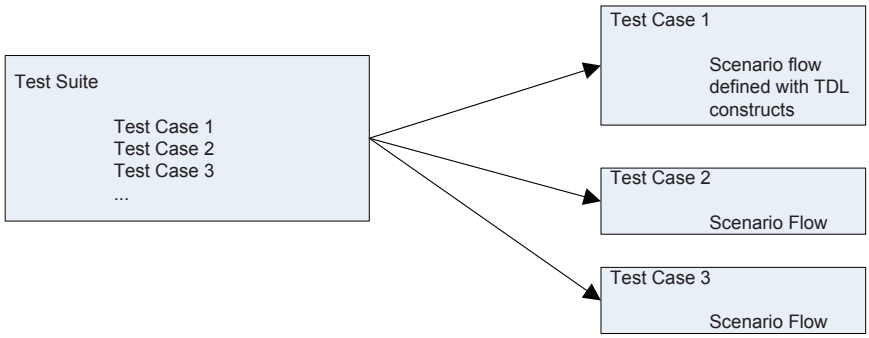

Figure 3.2: Test Suite - Test Case structure in TestBATN

Test suites are the containers for the test case definitions. Figure 3.3 presents the test suite definition of the Examination "Transmission Schema". It has a metadata field to indicate the title, version, maintainer, location and data of publish status and a short description about the test suite. Then, the definition of global variables show up which are accessible globally

from all test cases registered into the test suite. Afterwards, the test cases in that test suite are defined in order between the “TestCaseRefID” tags. The identifiers of the test case definitions are put inside these tags. The identifiers of the test cases should be unique in the context of the testsuite.

Within the scope of this thesis, all required testcases and related development such as Schematron [16] rules and XPATH [15] expressions to check the messages against the business rules are developed for several NHIS services, such as the Examination service which constitutes the Examination Test Suite. To enable the testing of the remaining HL7 services, these testcases and related definitions are extended and registered to the TestBATN framework as shown in Figure 1.1.

In the context of Examination test suite the following test cases are developed:

- Muayene_TemelTestSenaryosu: The basic conformance testcase which tests the minimal set of requirements to send a valid examination “Transmission Schema” instance.
- Muayene_TemelTestSenaryosu_Recete, Muayene_TemelTestSenaryosu_TetkikSonuc: - The customized basic conformance testcases which test the optional Prescription and Test Result Dataset fields, in addition to the tests done through the basic conformance testcase, inside an examination message respectively. Since through the basic conformance testcases, it is possible to send valid messages without these optional fields; customized versions of basic conformance testcases are highly required to enable the testing of the optional fields.
- Muayene_HastaCikisMSVSAnlamsalTestSenaryosu, Muayene_HastaKabulMSVSAnlamsalTestSenaryosu, Muayene_MuayeneMSVSAnlamsalTestSenaryosu, Muayene_ReceteMSVSAnlamsalTestSenaryosu, Muayene_TetkikSonucuMSVSAnlamsalTestSenaryosu: The functional and semantic testcases which test the functional capabilities of the client applications. Each semantic/functional testcase definition is specialized for the MHDSs which together form the examination “Transmission Schema”. By this means, semantic functionality detections are done for each MHDS separately inside a “Transmission Schema” instance.
- Muayene_TemelBirlikteIslerlikTestSenaryosu: The interoperability testcase which re-

```
<TestSuite xmlns="model.testsuite.testframework.srdc.com"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  id="Muayene_TestSuite">

  <MetaData>
    <Title>Saglik Bakanligi - Muayene Veri Akışı Testleri</Title>
    <Version>0.1</Version>
    <Maintainer>SRDC Team</Maintainer>
    <Location>Ankara,Turkey</Location>
    <PublishDate>24/03/2008</PublishDate>
    <Status>DRAFT</Status>
    <Description>Muayene MSVS Bildirimi Uyumluluk Testleri</Description>
  </MetaData>

  <Variables>
  </Variables>

  <TestCaseRefID>
    Muayene_HastaCikisMSVSAnlamsalTestSenaryosu
  </TestCaseRefID>
  <TestCaseRefID>
    Muayene_HastaKabulMSVSAnlamsalTestSenaryosu
  </TestCaseRefID>
  <TestCaseRefID>
    Muayene_MuayeneMSVSAnlamsalTestSenaryosu
  </TestCaseRefID>
  <TestCaseRefID>
    Muayene_ReceteMSVSAnlamsalTestSenaryosu
  </TestCaseRefID>
  <TestCaseRefID>
    Muayene_TetkikSonucuMSVSAnlamsalTestSenaryosu
  </TestCaseRefID>
  <TestCaseRefID>
    Muayene_SorgulamaServisiTestSenaryosu
  </TestCaseRefID>
  <TestCaseRefID>
    Muayene_TemelBirlikteIslerlikTestSenaryosu
  </TestCaseRefID>
  <TestCaseRefID>
    Muayene_TemelTestSenaryosu
  </TestCaseRefID>
  <TestCaseRefID>
    Muayene_TemelTestSenaryosu_Recete
  </TestCaseRefID>
  <TestCaseRefID>
    Muayene_TemelTestSenaryosu_TetkikSonuc
  </TestCaseRefID>
  <TestCaseRefID>
    Muayene_IptalServisiTestSenaryosu
  </TestCaseRefID>
  <TestCaseRefID>
    Muayene_GuncellemeServisiTestSenaryosu
  </TestCaseRefID>
</TestSuite>
```

Figure 3.3: Test Suite definition of Examination

quires the participation of the NHIS server and a FMIS/HIS application. The interoperation capabilities of the two systems are tested in the context of the choreography of the messages exchanged between the parties. The developed interoperability test-case makes TestBATN behave as a proxy between the communicating parties, listen the

exchanged messages and apply the necessary tests on them.

- Muayene_GuncellemeServisiTestSenaryosu: The update method of the examination service is simulated and the messages are tested against the corresponding update rules.
- Muayene_IptalServisiTestSenaryosu: The delete method of the examination service is simulated and the messages are tested against the corresponding deletion rules.
- Muayene_SorgulamaServisiTestSenaryosu: The query method of the examination service is simulated.

The developed testcases formed a basis for the development of testcases for all Web services published by NHIS.

In the following, descriptions of the test steps in each test scenario category are analyzed in detail by going over example testcase definitions. Complete testcase definitions with the corresponding Schematron definitions are presented in "http://www.srdc.metu.edu.tr/ anil/-MOH_Muayene.zip".

Basic Conformance Testcases

For each "Transmission Schema", a basic conformance testcase is written to test the conformance of FMISs and HISs to the requirements defined in the Integration Profile for the corresponding transmission. The following test steps describe these scenarios and the corresponding TestBATN framework functionalities used to execute them.

SOAP Message Conformance

The first step is the messaging step where the Systems Under Test (SUTs), the FMISs and HISs, are requested to send a transmission to the specified TestBATN Engine ports (TestBATN proxy ports as shown in Figure 1.1). By using the TestBATN messaging capability to receive the message from the client application with the "ReceiveMessage" and choosing the "SOAP Message Adaptor" (the adaptor approach in the TestBATN is describen in section 2.5) for this messaging step the scenario is configured to accept only valid SOAP messages. To handle the transportation, HTTP adaptor is used in the transport level to indicate the acception of the SOAP messages transported through HTTP. "SOAP Message Adaptor" and the "HTTP Message Adaptor" exist in the TestBATN framework as built-in adaptors, named as "SOAPReceiver" and "HTTPReceiver" respectively.

```
<ReceiveMessage cpaID="Etkilesim1" fromActor="USBSIstemci" id="R01"
  description="Bu Test Senaryosu sürecinde oluşan Muayene Bildirimini,
  USBSSimulator'ünün konfigürasyon kısmında belirtilen ağ adresine gönderiniz."
  transactionID="T01">
  <Transport transportHandler="HTTPReceiver"/>
  <Packaging packagingHandler="SOAPReceiver"/>
  <VariableRef>soapHeader</VariableRef>
  <VariableRef>soapBody</VariableRef>
</ReceiveMessage>
```

Figure 3.4: ReceiveMessage portion of the basic testcase for examination

Figure 3.4 shows the “ReceiveMessage” part from the examination basic testcase definition. After the arrival of the message to the TetBATN proxy ports, they are processed through the indicated adaptors and kept in the pointed variables.

Syntactic Validation of “Transmission Schemas”

This step is configured to use the built-in validation adaptor, “XSD Validation Adaptor”. Required inputs for this adaptor are the XML Schema of the corresponding transmission and the “Tranmission Schema” instance, that is the EHR message, received from the SUT (FMIS or HIS) in the SOAP Body.

```
<ValidateContent validationAdapter="xsdadapter"
  schemaLocation="testsuites/MOH_Muayene/MoHServer/XSDSchemas/muayene/MCCI_IN000001TR01.xsd"
  description="Gönderilen Muayene Bildirimi sözdizimsel yapısının belirlenen Muayene XML
  Semasına göre test edilmesi">
  <Expression>$MSVSbildirimi</Expression>
</ValidateContent>
```

Figure 3.5: Usage of XSD Adaptor

Figure 3.5 illustrates the usage of the “XSD Validation Adaptor” inside the “ValidateContent” construct. It gets the XML Schema of the examination service by the “schemaLocation” attribute and the other input, the message sent by the SUT is passed inside the variable “$MSVSbildirimi”. The adaptor checks the syntax of the message according to the provided XSD schema and outputs the result and the report of the validation process.

WS-Security Username - Token Profile Conformance

The “SOAP Message Adaptor” partitions the message into three fragments; HTTP Headers, the SOAP header and the SOAP body, and TestBATN framework allows the test designer to use these fragments independently in the proceding test steps by keeping them in the variables.

WS-Security Username - Token Profile and further restrictions defined in the Integration Profile regulate the usage of “UsernameToken” header in the SOAP Header. In order to test these restrictions, semantic validation steps of the TestBATN framework are used. Each semantic validation step is actually the evaluation of an XPATH expression (corresponding to a single restriction) over TestBATN built-in “XPATH Validation Adaptor”. For example, an XPATH expression is written to test whether the message includes a single “Username” element in “UsernameToken” header element. Figure 3.6 illustrates a validation step with XPATH expressions to check the existence of wsse:Password element in wsse:UsernameToken security header.

```
- <VerifyContent verificationAdapter="xpathadapter"
    description="'wsse:UsernameToken' elemanı bir tane wsse:Password
    elemanı içermelidir, ve bu elemanın 'Type' alanı 'http://docs.oasis-
    open.org/wss/2004/01/oasis-200401-wss-username-token-
    profile-1.0#PasswordText' olmalıdır.">
    <Expression>$soapHeader</Expression>
    <Expression namespaceBindings="{soap =
      http://schemas.xmlsoap.org/soap/envelope/, wsse =
      http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
      wssecurity-secext-1.0.xsd, wsu = http://docs.oasis-
      open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
      1.0.xsd}">count
      (/soap:Header/wsse:Security/wsse:UsernameToken/wsse:Password
      [@Type = 'http://docs.oasis-open.org/wss/2004/01/oasis-
      200401-wss-username-token-profile-1.0#PasswordText']) =
      1</Expression>
  </VerifyContent>
```

Figure 3.6: Verification Step for Username - Token Conformance

Testing the conformance of a SOAP message to the WS-Sercurity Username - Token Profile includes the checking of the wsse:Username and wsse:Password elements inside the header of the SOAP message. The test checks whether these elements exist on the right place of the SOAP header or not.

Username and Password Validation

The TestBATN user-interaction ability through TestBATN Control and Monitoring Environment (Chapter 4) enables the test designer to obtain some preliminary information from the

SUT administrator before test execution. Preliminary data is requested from the SUT administrator by defining the appropriate “PreliminaryData” instances through TDL.

```
<PreliminaryData toActor="USBSIstemci" description="Sisteminizin bu test için
  kullanacağı kullanıcı adı bilgisini giriniz." type="ask">
  <VariableRef>KullaniciAdi</VariableRef>
</PreliminaryData>
<PreliminaryData toActor="USBSIstemci" description="Sisteminizin bu test için
  kullanacağı şifre bilgisini giriniz." type="ask">
  <VariableRef>Sifre</VariableRef>
</PreliminaryData>
```

Figure 3.7: Preliminary Data Definition within TDL

Figure 3.7 illustrates the definitions of two preliminary data for the use in the Username and Password Validation. By means of this construct, before the testcase execution starts, the system will request two pieces of information as preliminary data from the user. This data will be kept in the variables “KullaniciAdi” and “Sifre” indicating username and password infomation which will be used in the SOAP Header through the WS-Security Username-Token Profile. That is, the SUT administrator tells the system which username and password information he or she will embed into the examination message that his or her system will send to the TestBATN framework in the proceding test steps. After obtaining the username and password information by processing the SOAP header, two semantic validation steps with XPATH Validation, the username and password provided in preliminary steps are compared with the values given in the “UsernameToken” field of the message. This test is used for checking the ability of the SUT to use the username and password information correctly by means of the marshalling of the message.

Validation of Coding Schemes and Codes

Health Coding Reference Server (HCRS) maintains all coding systems and codes in use in Turkey. As mentioned in section 2.4 and also at the beginning of this chapter, this coding information is available to the field through several Web services published within HCRS. All messages coming from the FMISs and HISs include parts to be represented with these well-defined coding systems and the code values. Figure 3.8 shows the diagnosis section of an Examination transmission. There is a “code” and a “value” field inside the “diagnosis” element. In the figure, these values together indicate that the patient in the subject of this

transmission has “K55” as “ANATANI”. The software systems need a mechanism to extract the semantics from these indications. Coding schemes and corresponding values exist for this reason. Inside the code system pointed with “2.16.840.1.113883.3.129.2.2.6”, “ANATANI” has a meaning and HCRS provides this meaning to all its clients through the Web services. Similarly, the value of “ANATANI” is indicated with “K55”. This value also has a meaning in the coding scheme pointed with “2.16.840.1.113883.6.3”. It actually means “BARSAĞIN VASKÜLER BOZUKLUKLARI”.

```
<diagnosis moodCode="EVN" classCode="OBS">
  <code code="ANATANI"
    codeSystem="2.16.840.1.113883.3.129.2.2.6"
    codeSystemName="Tanı Tipi" codeSystemVersion="1.0"
    displayName="Ana Tanı" />
  <value code="K55" codeSystem="2.16.840.1.113883.6.3"
    codeSystemName="ICD-10" codeSystemVersion="1.0"
    displayName="BARSAĞIN VASKÜLER BOZUKLUKLARI" />
</diagnosis>
```

Figure 3.8: Diagnosis section inside an Examination transmission

In order to validate the codes and code systems used in a transmission, the SKRS Validator adaptor is used. The SKRS Validator examines all data elements with coded value type from the transmission and calls the HCRS services to validate these code values and the corresponding code systems.

Figure 3.9 illustrates the usage of the SKRS validation phase in the definition of examination basic conformance testcase. “Assign” construct enables the processing of the XPATH expression to extract all “code” and “codeSystem” values from a transmission instance and assign these values to a variable, named “hl7CodeElementList”. All testing steps are dictated inside the “TestAssertion” construct of Test Description Language (TDL). Since SKRS adaptor is registered as a verification adaptor to the TestBATN adaptor system, it is executed through “VerifyContent” constructs.

SKRS adaptor has two versions, namely “SKRSALLValidator” and “SKRSValidator”. “SKRSValidator” examines all the code values and corresponding codeSystems passed inside the variable “hl7CodeElementList”. On the other side, there exist further restrictions on the transmissions defined in the Integration Profile. For instance, all messages arriving to the HL7 Web

```
<Assign id="ASGN_02" variableRef="hl7CodeElementList" append="no_append">
  <Expression namespaceBindings="{hl7 = urn:hl7-org:v3}">$MSVSbildirimi//*[@code]
    [@codeSystem]</Expression>
</Assign>
<TestAssertion id="TA3" description="Gönderilen Muayene Bildiriminde geçen tüm HL7 code
  referanslarının SağlıkNET SKRS sistemine uygunluğunun test edilmesi.">
- <VerifyContent verificationAdapter="SKRSALLValidator" description="Gönderilen Muayene
    Bildiriminde geçen tüm HL7 code referanslarının USBS SKRS sistemine uygunluğunun test
    edilmesi.">
    <Expression>$hl7CodeElementList</Expression>
  </VerifyContent>
- <VerifyContent verificationAdapter="SKRSValidator" description="Bildirimde geçen 'Kurum'
    kodunun SKRS sistemine uygunluğunun test edilmesi.">
    <Expression namespaceBindings="{hl7 = urn:hl7-
      org:v3}">'2.16.840.1.113883.3.129.1.1.6'</Expression>
    <Expression namespaceBindings="{hl7 = urn:hl7-
      org:v3}">$MSVSbildirimi//hl7:representedHealthcareOrganization/hl7:id/@extension</Expression>
  </VerifyContent>
```

Figure 3.9: Usage of SKRS Validator in TDL

services must have a field to identify the organization that is the owner of that message. Each healthcare organization in Turkey has a unique identifier, and these identifiers are embedded inside the messages. The validity of these identifiers should also be tested. “SKRSValidator” version of the adaptor takes a coding scheme and a code value as inputs and checks whether the value is valid within the scope of the provided coding scheme by communicating with the HCRS Web services.

Validation Against Business Rules

As mentioned earlier, for each transmission schema and MHDS, several business rules are defined to provide consistency among the values used in the data elements. An example of a Business Rule defined for the examination transmission schema is as follows:

“The ending time of the examination must be earlier than or equal to the discharge time of the patient indicated inside the discharge dataset”

Schematron definitions are used in specifying the business rules defined for the local constraints. For each transmission and for each MHDS in the transmission, the schematron rules of the corresponding business rules for that transmission or MHDS are defined. Figure 3.10 presents a scehmatron rule used in the testing of the business rule given above. The reporting mechanism of the schematron rules realize a good combination with the reporting mechanism of TestBATN framework. Through the TestBATN Control and Monitoring Environment these

reports are presented to the SUT users.

```
<sch:pattern name="rule 2 validation">
- <sch:rule context="hl7:examinationSection//hl7:encounter/hl7:effectiveTime">
  - <sch:assert test="srdc:isBeforeOrEqual(string(hl7:high/@value), string(root
      ()//hl7:dischargeSection//hl7:discharge/hl7:effectiveTime/@value))">
      Kural 2 geçerli değil (Muayene bitiş zamanı alanı, hasta çıkış msvs'deki çıkış
      zamanı alanından küçük ya da o tarihe eşit olmalıdır): Muayene Bitiş Zamanı:
      <value-of select="string(hl7:high/@value)" />
      - Çıkış Zamanı:
      <value-of select="string(root
        ()//hl7:dischargeSection//hl7:discharge/hl7:effectiveTime/@value)" />
    </sch:assert>
    <sch:report test="srdc:isBeforeOrEqual(string(hl7:high/@value), string(root
      ()//hl7:dischargeSection//hl7:discharge/hl7:effectiveTime/@value))">Kural 2
      geçerli (Muayene bitiş zamanı alanı, hasta çıkış msvs'deki çıkış zamanı alanından
      küçük ya da o tarihe eşit olmalıdır)</sch:report>
  </sch:rule>
</sch:pattern>
```

Figure 3.10: A Schematron Rule to test a Business Rule

To execute the developed schematron rules for each transmission schema and each MHDS inside the transmissions, “schematronadaptor” is used. “schematronadaptor” is another verification adaptor that comes built-in with TestBATN. Figure 3.11 illustrates the portion of the examination basic conformance testcase definition. Each set of schematron rules, corresponding to the business rules provided for the transmission and for each MHDS inside the transmission locally, are embedded inside the testcase definitions with “VerifyContent” construct of TDL inside a “TestAssertion”.

Interoperability Testcases

Basic conformance test scenarios cover the most of the testing steps. However, only testing the FMISs’ and HISs’ conformances to the Integration Profile does not guarantee that these applications will successfully interoperate with the NHIS server. For this reason, interoperability test scenarios are also needed to enable the comprehensive testing. The developed interoperability testcase for the examination transmission is presented inside the bundle in “http://www.srdc.metu.edu.tr/ anil/MOH_Muayene.zip”.

The approach of the interoperability testcases are as follows:

- According to the testcase definition, TestBATN functions as a proxy between the SUTs (FMISs and HISs) and the MoH NHIS Server. The “Transmission Schema” instances

```
<TestAssertion id="TA4" description="Gönderilen Muayene Bildiriminin, bu bildirim için belirlenen iş kuralları göz
  önüne alınarak test edilmesi">
- <VerifyContent verificationAdapter="schematronadapter" description="Gönderilen Muayene Bildiriminin, bu
    bildirim için belirlenen iş kuralları göz önüne alınarak test edilmesi">
    <Expression>$MSVSbildirimi</Expression>

      <Expression>'testsuites/MOH_Muayene/MoHServer/Schematrons/MuayeneSchematronNs.xml'</Expression>
  </VerifyContent>
- <VerifyContent verificationAdapter="schematronadapter" description="Gönderilen Muayene Bildiriminin Muayene
    kısmının, bu bildirim için belirlenen iş kuralları göz önüne alınarak test edilmesi">
    <Expression>$MSVSbildirimi</Expression>
    <Expression>'testsuites/MOH_Muayene/MoHServer/Schematrons/MuayeneMSVS.xml'</Expression>
  </VerifyContent>
- <VerifyContent verificationAdapter="schematronadapter" description="Gönderilen Muayene Bildiriminin Hasta
    Kabul kısmının, bu bildirim için belirlenen iş kuralları göz önüne alınarak test edilmesi">
    <Expression>$MSVSbildirimi</Expression>
    <Expression>'testsuites/MOH_Muayene/MoHServer/Schematrons/HastaKabulMSVS.xml'</Expression>
  </VerifyContent>
- <VerifyContent verificationAdapter="schematronadapter" description="Gönderilen Muayene Bildiriminin Hasta
    Cıkış kısmının, bu bildirim için belirlenen iş kuralları göz önüne alınarak test edilmesi">
    <Expression>$MSVSbildirimi</Expression>
    <Expression>'testsuites/MOH_Muayene/MoHServer/Schematrons/HastaCikisMSVS.xml'</Expression>
  </VerifyContent>
- <VerifyContent verificationAdapter="schematronadapter" description="Gönderilen Muayene Bildiriminin varsa
    Reçete kısmının, bu bildirim için belirlenen iş kuralları göz önüne alınarak test edilmesi">
    <Expression>$MSVSbildirimi</Expression>
    <Expression>'testsuites/MOH_Muayene/MoHServer/Schematrons/ReceteMSVS.xml'</Expression>
  </VerifyContent>
```

Figure 3.11: Schematron Adaptor used in TDL

produced by the sender application are forwarded directly to the MoH Server by the TestBATN framework. Similarly, the transmission responses produced by the MoH Server are intercepted and then forwarded to the SUTs.

- The framework internally stores all intercepted messages for further testing. Syntactic and semantic validations of Communication, Document and Business Process layers are performed as done in the basic conformance testcases.

- According to the developed testcase, TestBATN framework generates its own transmission response based on the profile constraints described in the MHDS. TDL enables the usage of the constructs [69] necessary to emulate the desired application behaviour.

- As the final step, the transmission responses of the MoH Server are validated against the ones generated internally by the TestBATN framework. At this stage, any inconsistency is an indication of the MoH Server's noncompliance with the profiles and in such a case, convenient test reports are generated.

Semantic Testcases

The TestBATN framework enables run-time customization of the scenario templates by means of its interaction capabilities with the SUT administrators (administrators of the FMISs, HISs and the MoH Servers). The TestBATN Test Description Language (TDL) supports user-interaction schemes to be defined either by its PreliminaryTestData or RequestTestData element constructs [69]. In this manner, three different user-interaction schemes have been developed and used in the NHIS tests.

i Prior to scenario execution, the user is requested to fix the values for various test parameters:

Some of the testcases involve certain parameters to be fixed before the execution of the testcase by the SUT user. The TestBATN framework uses the constructs called preliminary test variables for this purpose. For example, the healthcare institution and the author of the CDA document engaged in the "Transmission Schema" instance may need to be fixed in advance so that later on, the framework can use this information to perform various semantic tests such as

- the interface's ability to properly place this information into the related parts of the "Transmission Schema" instance,
- the FMIS's or HIS's compliance with the NHIS business rules. There is no limitation as to how preliminary test variables can be used within a test scenario, thanks to the capabilities of the TestBATN Test Description Language [69].

ii Proir to testcase execution, the scenario designer fixes some parameter values to test SUTs ability to work in a certain mode of operation and with the given control parameters:

In contrast to the approach followed in (i), this time, the test designer imposes certain restrictions on how the SUT user should control its application behavior. The main objective behind this user-interaction scheme is to verify that the developed FMIS or the HIS is able to run in different modes of operation and that the application fulfills the requirements specific to that mode of operation. As an example, consider the case where the doctor fills out a detailed report regarding the referral of a patient to another healthcare institute. In this case, checking that the client application processes the user input and it places the correct ICD-10 code [61] is one such test alternative for such a scenario.

iii During the execution of the testcases, the user is requested to provide feedback on the observed application behaviour:

It is required to obtain feedback from the user during the testcase execution for the purpose of assessing the SUT's rendering capabilities. Therefore, the semantic testcases include the "ask" operation to ask a question to the SUT user at the time of the testcase execution. For example, the test semantic testcases may include a step; asking the admittance date/time of the patient in subject of the message that the SUT has received from the TestBATN proxy ports as a response of its activity.

Any HIS or FMIS can send valid messages to the NHIS services as long as the message is valid according to the syntactic and business rules declared in the Integration Profile. However, an important issue remains unchecked; the semantic/functional capabilities of the client applications are not tested. The quality of the message content cannot be assured only through the conformance and interoperability tests. For example, a HIS may create an instance of a valid message and send that message by only changing small portions of it. That is, we can never be sure whether the client application is capable of sending the correct values which are indicated by the users of it. A physician may want to send an examination "Transmission Schema" instance after examining the patient by writing some prescription. The physician may set several parameters about the prescription such as dose quantitiy, the name of the medication, period value and etc. We want to be sure that the underlying client application is capable of correctly setting the values of these parameters inside the message instance to be sent to the NHIS servers.

Figure 3.12 illustrates the mechanism of a semantic testcase. In the given portion, the testcase definition requests some fixed values from the SUT. This illustration constitutes an example of the semantic testcase capabilities of TestBATN described above as "Prior to testcase execution, the scenario designer fixes some parameter values to test SUTs ability to work in a certain mode of operation and with the given control parameters". According to the scenario requirements presented in Figure 3.12, the SUT must send a message which includes a prescription part and inside that part the medication must be "aspirin" with some additional fixed constraints. The corresponding fields are represented inside the message in Figure 3.12. As a result, by changing these fixed requirements with this mechanism, or adopting other semantic mechanisms available inside TestBATN, the client applications can be tested in a more

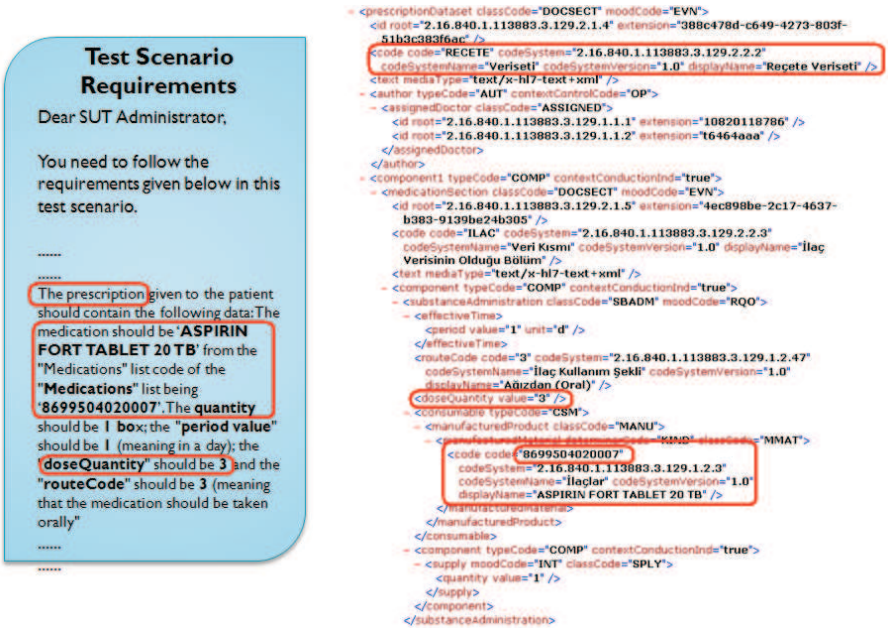

Figure 3.12: Semantic Test Scenario Example

comprehensive way.

Semantic and functional testing capabilities make TestBATN a good platform for the certification of applications developed within NHIS. Indeed, these kind of tests form the basis of the testing of the applications for certification purposes for any standard or standard group in any domain.

Testcases for Query/Update/Delete

NHIS published 25 HL7-based Web services. In the previous sections, the developed testcases target to the “Transmission Schema” instances for the “insert” operation of these services. However, the published Web services also contain “query”, “update” and “delete” operations. In the scope of this thesis, the testcases for “query”, “update” and “delete” methods of the Web services in question are also developed and registered to the TestBATN framework.

Each “Transmission Schema” instance to be sent to any service of NHIS must include a unique Clinical Document Identifier. “Insert” method accepts the messages and registers to the NHIS databases with the assigned identifier by the FMISs and HISs. The uniqueness of these identifiers comes from the “Universal Unique Indefier” concept that guarantees to

generate unique identifiers at the time of generation independent from the hosting machine, operating system etc. The remaining three methods of the service use the previously sent identifiers for the corresponding messages.

```
<replacementOf>
- <parentDocument>
    <id root="2.16.840.1.113883.3.129.2.1.3" extension="21302438-08ab-42c5-9c5e-
     17064c133456" />
    <code code="MUAYENE" codeSystem="2.16.840.1.113883.3.129.2.2.1"
      codeSystemName="Döküman Tipi" codeSystemVersion="1.0" displayName="Muayene
      MSVS (Vatandaş/Yabancı)" />
    <versionNumber value="1" />
  </parentDocument>
</replacementOf>
```

Figure 3.13: A Portion of an Update Message

For example, Figure 3.13 shows the related part of an update message to be sent to the examination service of NHIS. The update messages include a part named with “replacementOf” indicating that the message in question is going to update the message which is previously sent with the identifier which is indicated with the highlighted part in Figure 3.13.

Testcases for “query”, “update” and “delete” operations require an “insert” message which is sent previously. Since TestBATN only simulates the NHIS services, it does not maintain any database for the previously sent messages by the clients. Therefore, the developed testcases for these methods first request a valid message from the SUT to behave it as a message insertion prior to the intended operation, afterwards, request the query, update or delete message according to the executed testcase.

CHAPTER 4

TestBATN CONTROL AND MONITORING ENVIRONMENT

TestBATN is a set of tools enabling the comprehensive testing of all layers in the interoperability stack. To enable the testing of National Health Information System (NHIS) of Turkey, several testcases are developed in the scope of this thesis as described in the previous chapter. TestBATN Control and Monitoring Environment realizes a Web based graphical user interface and associated tools to enable the testing of the Family Medicine Information Systems (FMISs) and Hospital Information Systems (HISs), online.

Being Web based holds importance at this point from two different perspectives. First, a System Under Test (SUT) administrator does not desire to install software programs to test the conformance of the implementation. That is, for each standard and/or protocol available in the interoperability stack, the application developers may need to install different software tools, and compatibility issues arise with these several of tools.

On the other hand, interoperability tests imply a step-forward importance than on the comformance tests as described in Chapter 2. And, interoperation of a number of systems requires those systems to get together and send/receive messages to/from each other. Today, there are organizations in several sectors to bring these systems together "physically" within a series of workshops and test the interoperability capabilities of these systems. IHE [77] Connectathon [71] is a testing event to enable the healthcare IT industry's large-scale interoperability testing. In these events, vendors come together with their systems and test their abilities to interoperate with other vendor systems exist in the workshop event.

TestBATN framework overcomes this physical restriction by providing the TestBATN Control and Monitoring Environment as a Rich Internet Application (RIA). Software vendors can easily go online through TestBATN and test their interopration capabilities through the inter-

operability testcases deployed under the TestBATN framework. The environment provides a Flex [29] based RIA that can run in any major browser with a Flash Player [38] plugin. The RIA makes use of the SOAP Web services technology to communicate with the TestBATN database for retrieval of the testsuite and testcase definitions in addition to the user account, test reports and statistical processing operations. Furthermore, in a testing environment to enable the online communication, reporting and status information facilities, there is a need for fast communication architecture between the Systems Under Test (SUTs) and the TestBATN server, and between the multiple clients in an interoperability testcase. TestBATN Control and Monitoring Environment introduces its own XML [9] based protocol on the TCP [11] level to realize this high speed communication.

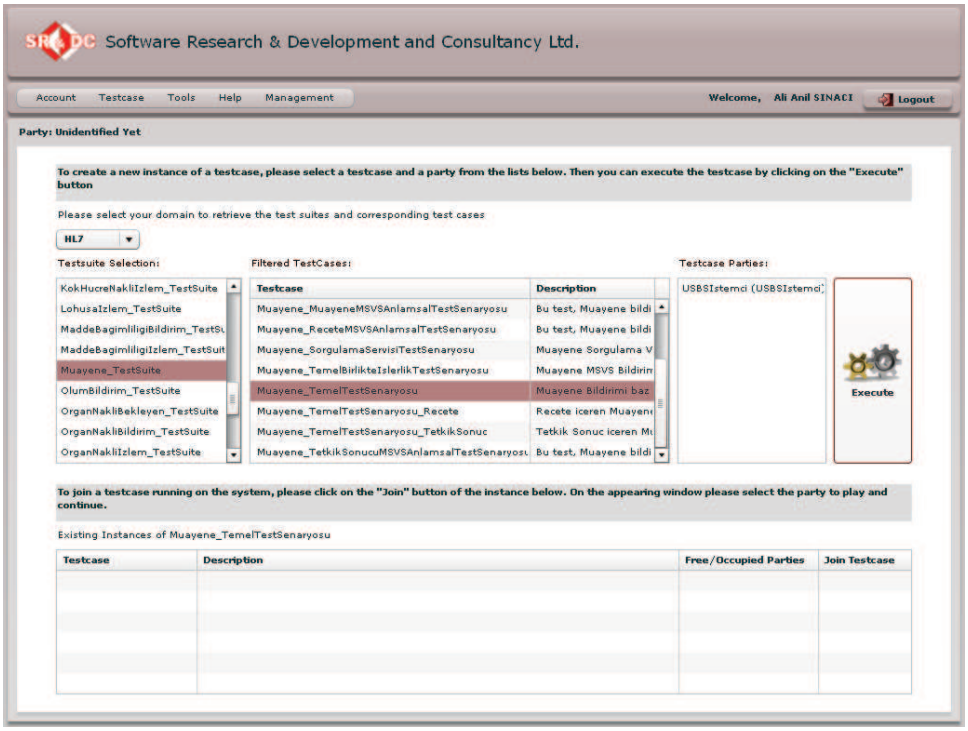

Figure 4.1: TestBATN Control and Monitoring Environment Mainpage

Figure 4.1 shows the main page of the RIA interface of the TestBATN Control and Monitoring Environment when any SUT administrator goes online.

Through the TestBATN Control and Monitoring Environment SUT administrators can do the following operations online and in high speed like on a desktop environment by deriving benefit from the capabilities of RIA facilities:

- login to the online framework and manage their account information.
- list the testsuites and corresponding testcases registered at the TestBATN database.
- filter the testsuites according to the application domain. For the testing of NHIS, the users select “HL7” domain. Then the testsuites registered for the testing of application in the “HL7” arena related to NHIS are listed on the testsuites field.
- filter the testcases according to the parent testsuite. When a user selects a testsuite, the related testcases are listed accordingly.
- retrieve the running testcase instances on the TestBATN framework. When a user selects a testcase on the mainpage, the running instances of that testcase are listed on the page. The purpose of this facility is to enable the interoperability testing of several systems through the registered interoperability testcases.
- join to any running testcase instance if the suitable party is not occupied and the joining is available.
- execute a testcase. This creates a new running instance of that testcase definition inside the TestBATN framework.
- enter the required information to handle the configuration management inside the framework for maintaining the required proxy ports etc.
- enter and view the preliminary information if defined through the selected testcase definition.
- view the steps of the testcase definition.
- start the execution of the testcase and monitor all the events online at the occurrence time with the help of the test driving protocol which is detailed in section 4.2.
- view and manage the questions asked according to the testcase definition with the “AskTestData” construct of the TestBATN Test Description Language.
- stop the execution of the testcase at any time.
- view the reports for each test step separately just after the reports arrive to the RIA interface through the test driving protocol.

4.1 General Architecture

Figure 4.2 presents the general architecture of the TestBATN Control and Monitoring Environment through the Test Driving Protocol perspective. TestBATN core framework has the convenient implementation of the protocol. The other end of the protocol is implemented in Flex as a Rich Internet Application. The RIA handles the operations which require high speed communication through the Test Driving Protocol and the remaining ones which are directly related with the retrieval of data from the TestBATN Database through the TestBATN Web Services.

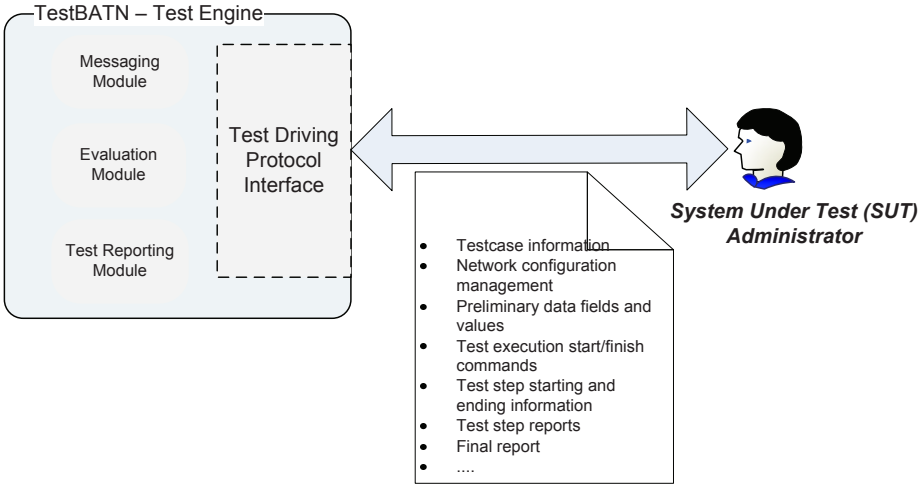

Figure 4.2: General Architecture of the Environment

RIA Implementation

The Web interface part of the TestBATN Control and Monitoring Environment is implemented in Adobe Flex. Flex provides rich constructs to create friendlier and interactive graphical user interfaces (GUI). Since the Flex outputs are Flash executables, namely SWF [39] files, every major browser including a Flash Player plugin can execute the GUI of the environment. Indeed, since Flash is a run-time environment with several features such as exception handling, socket communication facilities; the GUI does the most of the processing job. That is, the processing is mostly done on the browsers of the clients, thus the core of the TestBATN framework does not deal with heavy processing operations.

Throughout the implementation process, the Cairngorm [42] micro-architecture is adopted as a set of design patterns. For each construct inside the Test Driving Protocol, convenient data structures are created and maintained separately from the presentation and controller layers as the Cairngorm micro-architecture advices. The data model of the whole application is maintained through the singleton pattern as a singleton class.

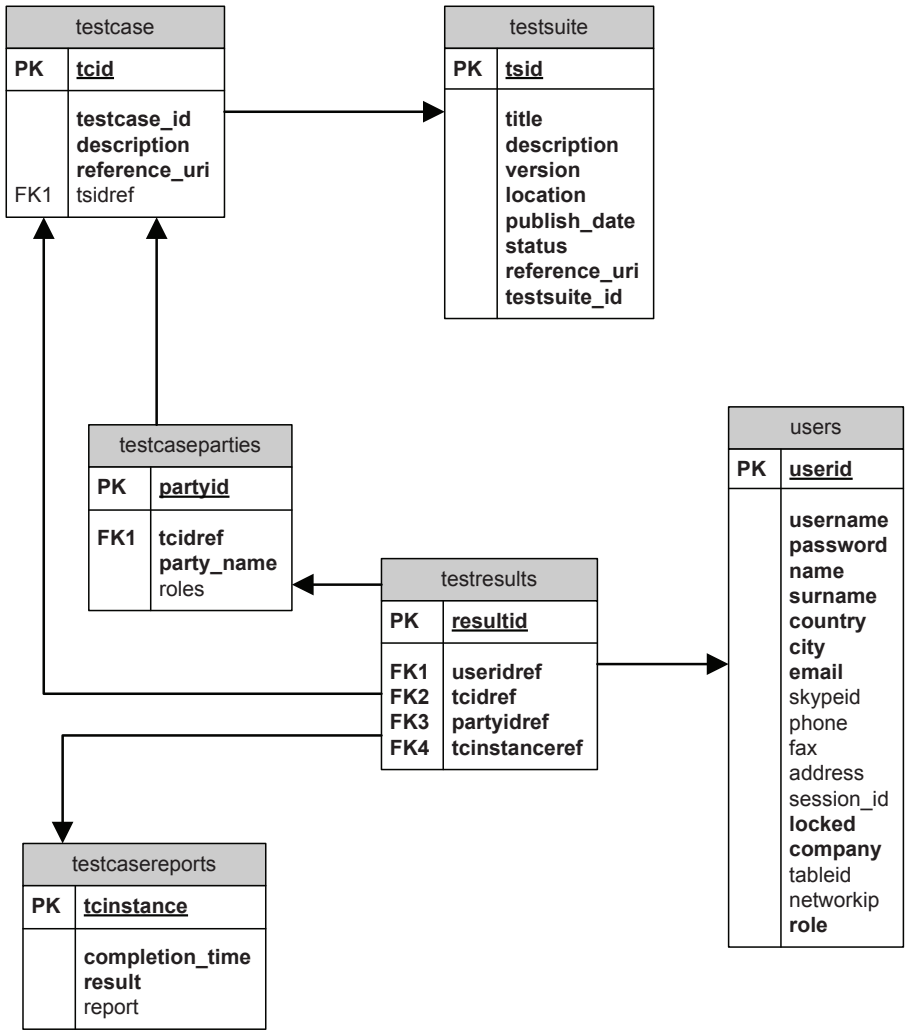

Figure 4.3: TestBATN Database ER Diagram

Binding mechanism of Flex is heavily used to realize the Cairngorm design paths, to bind the data structures inside the model to the presentation constructs such as text fields, checkboxes etc. By this means, when any change occurs in any data element residing inside the model, the corresponding presentation element is updated automatically through the binding mechanisms

built in Flex.

The communication operations are handled through another singleton class as named under business operations. The operations related with the retrieval of the testsuite/testcase data, user information and previously executed testcase results/reports are done through the Test-BATN Web services as described below. On the other side, some operations need speed in the context of the conformance and interoperability testing at the monitoring phase mostly. These operations are also described in detail in the Test Driving Protocol section.

TestBATN Database

TestBATN Web services are invoked mainly for direct interaction with the TestBATN Database. Figure 4.3 shows the Entitiy-Relationship diagram of the database. The relations of the database are designed to meet the requirements of the service functions serving to the Test-BATN Control and Monitoring Environment.

TestBATN Web Services

The communication with the TestBATN server is done through the TestBATN Web Services for the following operations as shown in Figure 4.4

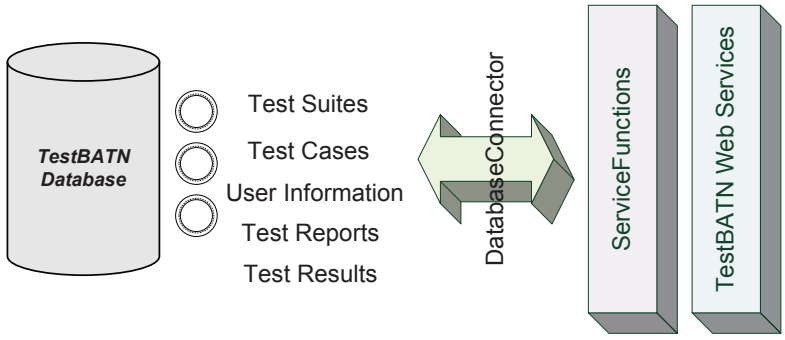

Figure 4.4: Interaction between the TestBATN Web Services and Database

- registerNewUser: Allows the registration of new users to the TestBATN framework. Only registered users are allowed to go online through the TestBATN Control and Monitoring Environment.
- checkCredentials: This service is invoked to check the username-password information of the SUT administrators to go online.

- updateUserInfo: The logged in users are able to update their account information through this service.

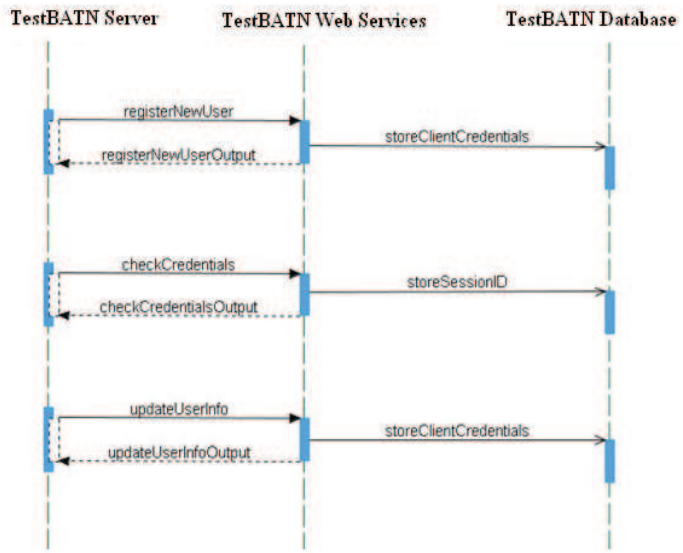

Figure 4.5: registerNewUser, checkCredentials, updateUserInfo

- getTestSuites: When the SUT administrator goes online, this method is invoked to retrieve the testsuites registered to the system.

- getTestcases: This is invoked just after the "getTestSuites" operation to retrieve the all registered testcases.

- getTestcaseParties: All testcase definitions include a number of parties to be actored when the testcase is executed. In the NHIS case, the FMIS/HIS administrators select the "USBSIstemci" actor to indicate that they are the clients of NHIS at the testcase in execution. This service retrieves the party information for each testcase to present to the SUT administrator for enabling the selection of the appropriate role before the execution of the testcase.

- getTestResults: SUT administrators are able to see their previously executed test results. This service provides the required functionality to enable this facility.

- getTestCaseReport: SUT administrators are able to retrieve their previously executed testcases' reports through the invocation of this method.

Figure 4.6: getTestSuites, getTestCases, getTestCaseParties, getTestResults, getTestCaseReport

- getAllUsers: This is an administrative method which is invoked when an a TestBATN administrator wants to see the all registered users.

- queryTestStatisticsForCompany: This method is invoked to retrieve statistical information on the testcase executions of the registered SUTs.

- queryTestResultsForTestSuite: Another statistics purposed method to retrieve testresults in the context of successes and failures for each testsuite.

Figure 4.7: getAllUsers, queryTestStatisticsForCompany, queryTestResultsForCompany, queryTestResultsForTestSuite

4.2 Test Driving Protocol

TestBATN Control and Monitoring Environment implements a Test Driving Protocol at the TCP level through direct socket communication to enable execution and management of test-cases, monitoring of the test steps through the execution and additional administrative operations. The protocol is XML based and has an XSD Schema which is described in detail in this section.

The capabilities provided by the Test Driving Protocol can be grouped under three categories which are described in the following subsections.

4.2.1 Initiation and Termination of SUT Sessions, and Monitoring the Already Running Testcase Instances

SUT administrator goes online to the main page of the TestBATN Control and Monitoring RIA through the appropriate web service calls. When the login operation is successful, the Test Driving Protocol starts functioning by initiating a session on the TestBATN server application. Figure 4.8 illustrates the sequence of the related message chroeography throughout the session management.

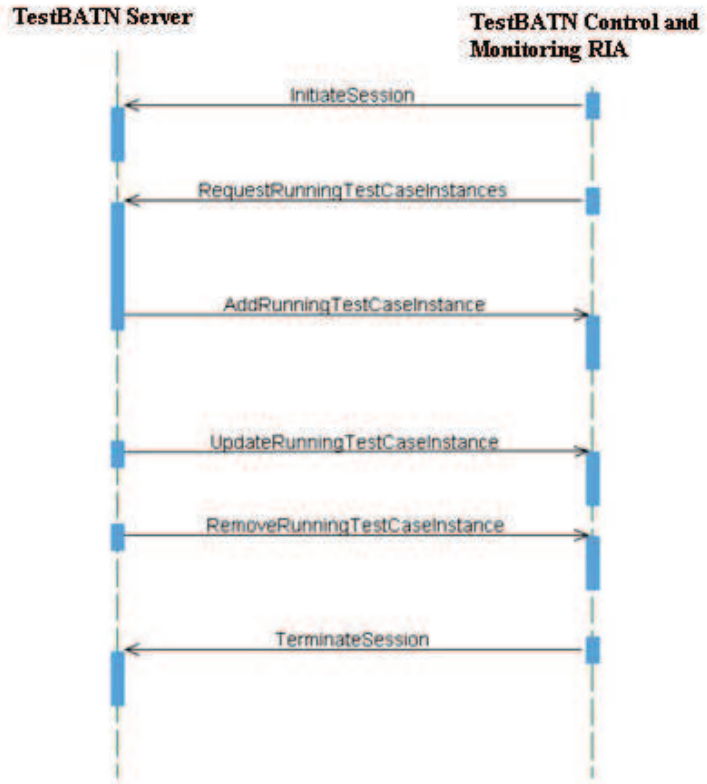

Figure 4.8: Management of the Session

Figure 4.9 shows the snapshot from the TestBATN Control and Monitoring Environment.

As seen from the figure, when the user selects the “Muayene_TestSuite”, the testcases defined in examination testsuite are listed as “Filtered TestCases”. If the user selects the “Muayene_TemelBirlikteIslerlikTestSenaryosu”, which is the basic interoperability testcase for the NHIS examination service, the already running instances of that testcase are listed below, at the sec-

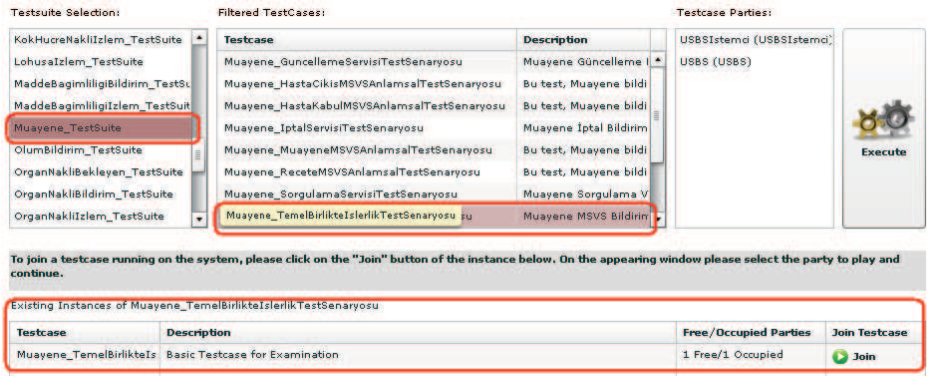

Figure 4.9: Already Running Testcase Instances

tion “Existing Instances of Muayene_TemelBirlikteIslerlikTestSenaryosu”. The interactions which enable these capabilities, which take part in the choreography, are described in detail in the following subsections.

Interaction: InitiateSession

Description:

Notifies the TestBATN server when a SUT administrator goes online. All remaining interaction is handled over this obtained session. This message is the handshake portion of the choreography. The Authenticator field inside the message is handshaked by the both sides and is sent inside all other messages exchanged between them.

Figure 4.10: Model of InitiateSession

Sender Responsibilities:

TestBATN Control and Monitoring RIA verifies the credentials of the user before sending this message.

Receiver Responsibilities:

TestBATN server cross-checks the session related information sent through this message with the internal mechanisms for security reasons.

Repeatability:

No; it cannot be repeated until the session is terminated.

Interaction: RequestRunningTestCaseInstances

Description:

Before the execution of any testcase, when a user clicks on a testcase definition all the running instances of that testcase are listed on the main page. This message indicates the request that the SUT administrator wants to see the running instances of the testcase.

Sender Responsibilities:

TestBATN Control and Monitoring RIA sends this message to the server upon to the user's appropriate action in the testcase selection.

Receiver Responsibilities:

TestBATN server provides the running instances of the testcase with "AddTestCaseInstance" response when it receives this message. Until another "RequestTestCaseInstances" message arrives to the server, it continuously updates (with "AddRunningTestCaseInstance", "UpdateRunningTestCaseInstance" and "RemoveRunningTestCaseInstance" messages) the RIA when any change occurs about the running instances of the testcase which is requested lastly by the RIA.

Repeatability:

Yes; it can be sent to the TestBATN server at each user request.

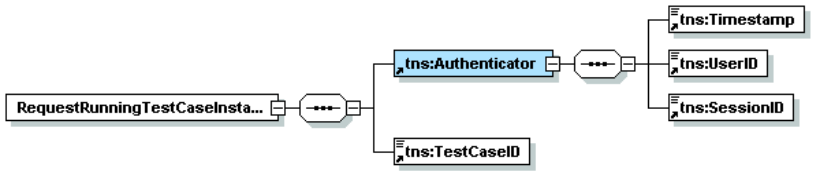

Figure 4.11: Model of RequestRunningTestCaseInstances

Interaction: AddRunningTestCaseInstance

Description:

This message construct is used after the RIA sends “RequestRunningTestCaseInstances” message to the TestBATN server. The server sends “AddRunningTetCaseInstance” message to the RIA if any instance of the testcase indicated by the lastly sent “RequestRunningTestCaseInstance” message is created by any other SUT administrator online in the TestBATN framework.

Sender Responsibilities:

TestBATN server prepares the content of the message, which is the information about the running instance of the testcase, and sends to the RIA client.

Receiver Responsibilities:

The RIA, that is the GUI running in the browser of the SUT administrator, updates the presentation of the related parts when any “AddRunningTestCaseInstance” message arrives.

Repeatability:

Yes. This message can be sent from the TestBATN server to the RIA when any testcase instance is started execution.

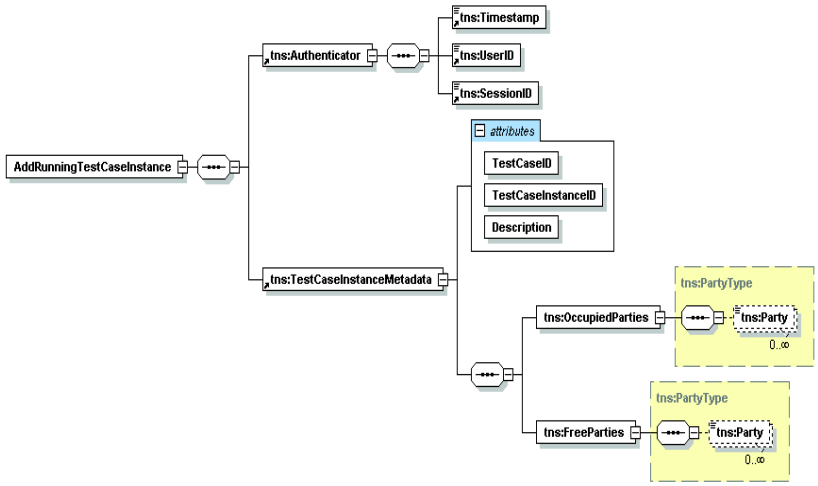

Figure 4.12: Model of AddRunningTestCaseInstance

Interaction: UpdateRunningTestCaseInstance

Description:

After the realization of the interaction, “RequestRunningTestCaseInstances”, the TestBATN server sends this message to the RIA when any change occurs on the testcase execution instace which was previously sent to the RIA client by the “AddRunningTestCaseInstance” message. For example, if a new SUT attends to a running testcase instance by occupying a testcase party, this change is sent to the RIA clients who previously sent “RequestRunningTestCaseInstances” to the TestBATN Server about the testcase in question.

Sender Responsibilities:

TestBATN server sends this message to the corresponding RIA clients when any change occurs on the executing testcase instances.

Receiver Responsibilities:

The RIA updates the corresponding GUI part when receives this message.

Repeatability:

Yes; the server sends this message for each change on the executing testcase instance.

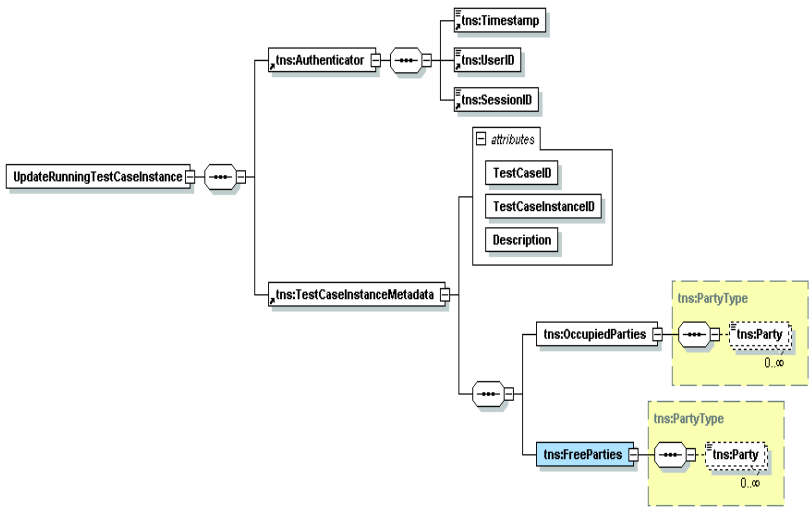

Figure 4.13: Model of UpdateRunningTestCaseInstance

Interaction: RemoveRunningTestCaseInstance

Description:

This message is sent from the TestBATN server to the RIA clients when any test-case execution terminates. The server sends the message to the RIA clients who previously sent “RequestRunningTetCaseInstances” for a testcase definition.

Sender Responsibilities:

When a testcase ends up, the server sends this message to the RIA clients who previously sent “RequestRunningTestCaseInstances” message for that testcase.

Receiver Responsibilities:

The RIA updates the necessary parts of the GUI upon reception of this message.

Repeatability:

Yes; when any running testcase instance terminates, the message is sent to the corresponding RIA clients. Creation of this message is agnostic to the reason behind the termination of the testcase.

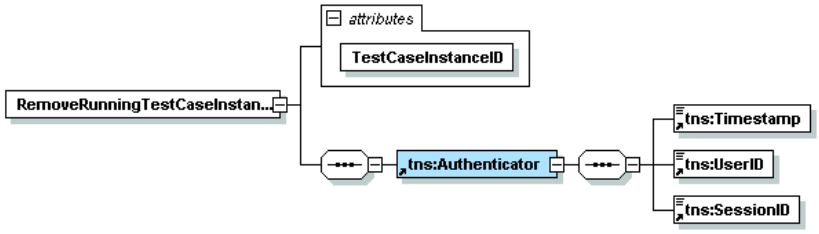

Figure 4.14: Model of RemoveRunningTestCaseInstance

Interaction: TerminateSession

Description:

This message is sent from the RIA client to the TestBATN server when the SUT administrator wants to logout. It can be seen as the opposite of the “InitiateSession” message for dropping the handshake between the both sides.

Sender Responsibilities:

Test Control and Monitoring Environment must send the message through the RIA client to the TestBATN server for each logout process.

Receiver Responsibilities:

The server clears the session information from the internal system and handles the configuration parameters of the user and the testcases executed by that user.

Repeatability:

No.

Figure 4.15: Model of TerminateSession

4.2.2 Test Execution

SUT administrator selects the testcase party to actor through the testcase after selecting the testcase from the “Filtered Testcases” section as shown in Figure 4.16. Afterwards, the user clicks on the “Execute” button to proceed to start the testcase execution.

Figure 4.16: Testcase Selection and Execution

According to the underlying Test Driving Protocol, when the user clicks on the “Execute” button, the RIA sends “LoadTestCase” interaction message to the TestBATN server. Figure 4.17 presents the messaging choreography between the TestBATN Control and Monitoring Environment RIA and the TestBATN server starting from loading a testcase and ending with the start of the execution of the testcase. In the following sections, the steps of this chorepgrahy are described and the intermediate interactions are explained in detail.

Interaction: LoadTestCase

Description:

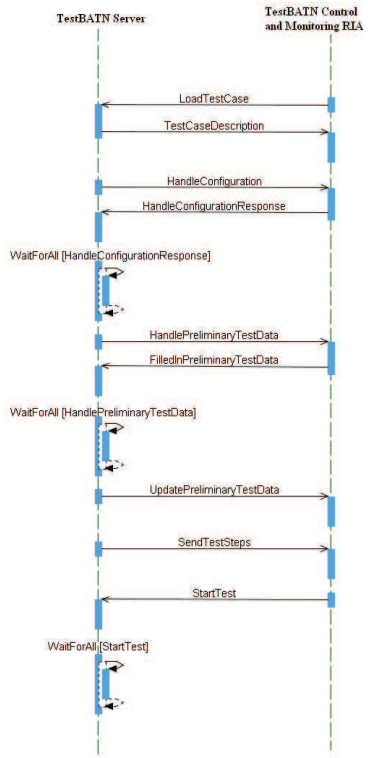

Figure 4.17: Execution of a Testcase

This message is sent from the RIA to the TestBATN server to load a testcase to be executed.

Sender Responsibilities:

RIA sends this message to the server when the SUT administrator selects the testcase and presses the “Execute” button from the GUI.

Receiver Responsibilities:

TestBATN server loads the specified testcase to the memory and sends the description of the testcase and the configuration information to the RIA.

Repeatability:

No. It cannot be repeated until the RIA sends the “ReleaseTestCasePorts” message.

Upon receiving a “LoadTestCase” message, the server sends the description of the loaded testcase, “TestCaseDescription”, to the RIA client. The RIA pops up a new window and shows the description of the testcase to the SUT administrator. This description includes the testing steps of the loaded testcase and their brief explanations.

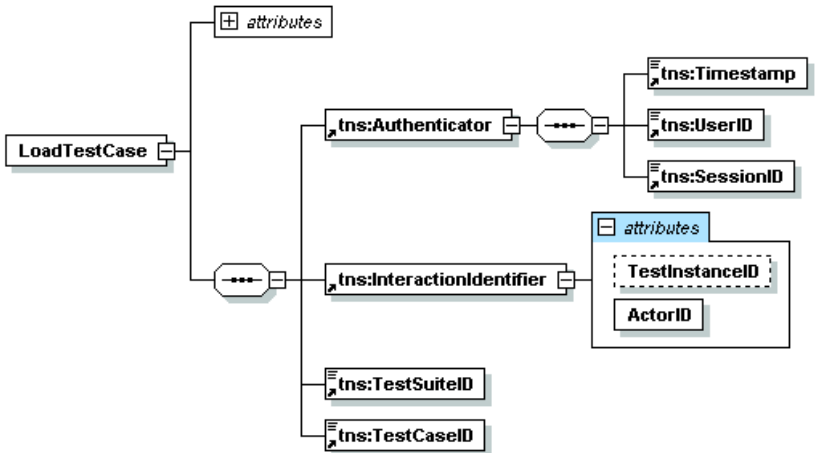

Figure 4.18: Model of LoadTestCase

Interaction: TestCaseDescription

Description:

TestBATN server sends the description of the testcase to be executed to the RIA.

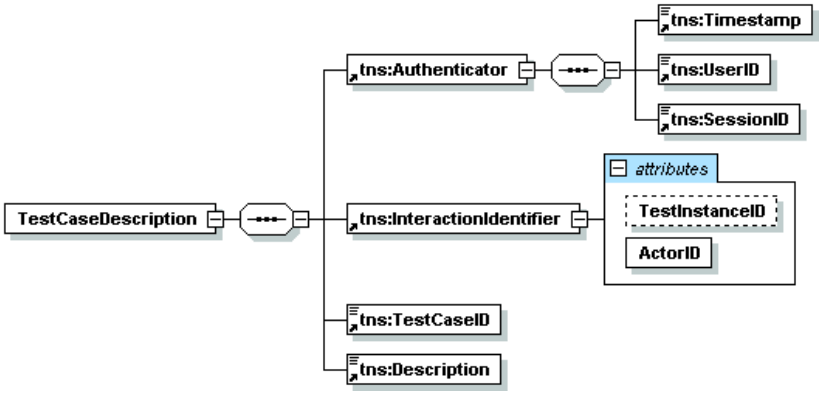

Figure 4.19: Model of TestCaseDescription

Sender Responsibilities:

TestBATN server should prepare the testcase for the execution, assign a unique identifier to the execution instance and send this information to the RIA.

Receiver Responsibilities:

RIA should present the description to the SUT administrator. The SUT administrators not only see the steps of the testcase but also learn the details and rules that they must apply into their messages. This process increases the awareness of the FMIS/HIS developers about the standards and specifications dictated through the Integration Profile.

Repeatability:

No.

Right after sending the “TestCaseDescription”, the server sends “HandleConfiguration” to the RIA to configure the network related details between the parties of the testcase.

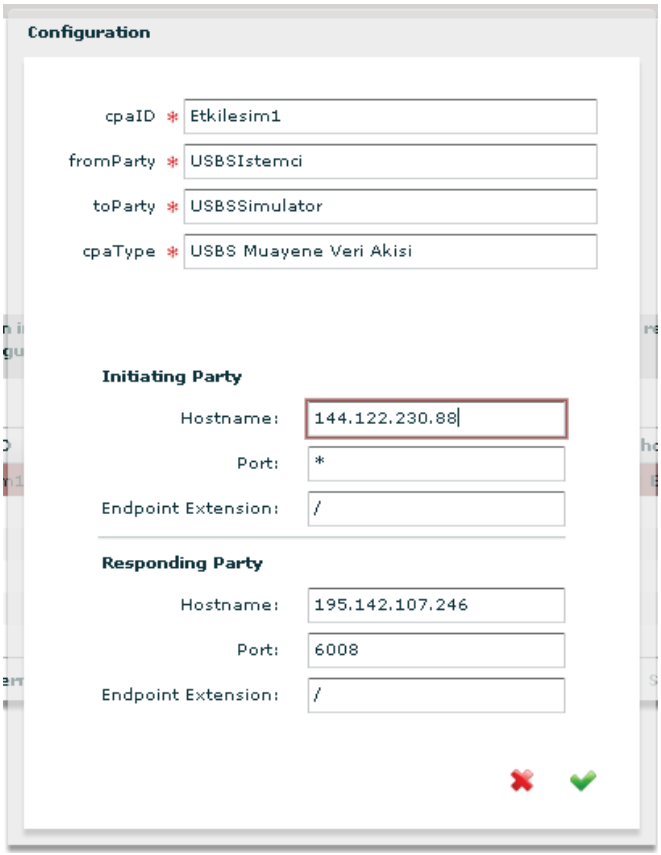

Figure 4.20: Handling Configuration

TestBATN Control and Monitoring Environment RIA lists the “HandleConfiguration” information defined inside the testcase. RIA presents this information in a user-friendly GUI window to the SUT administrator as shown in Figure 4.20. For each communication that

will occur through the testcase execution, there must be a configuration handling part defined inside the testcase definition. This configuration includes the management of the IP (Internet Protocol) addresses and the port numbers of the parties taking role through the testcase. That is, in Figure 4.20, the SUT administrator only enters its IP address information as the Initiating Party because, for that case, the SUT administrator plays the “USBSIstemci” role which is the NHIS client. The Responding Party is the NHIS simulation, which is the TestBATN server. The port number of the TestBATN is important since the SUT administrator must configure its software to send the “Transmission Schema” instance to the specified port number on the IP address of the TestBATN server, the Responding Party.

Interaction: HandleConfiguration

Description:

For all communication lines defined in the testcase, there must be corresponding configuration information. TestBATN server asks this configuration information from the SUT administrator by means of this interaction.

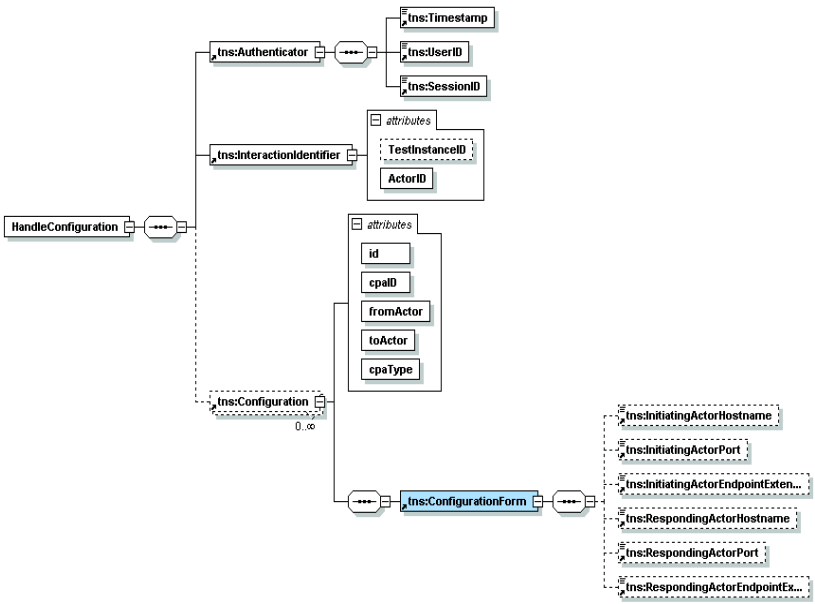

Figure 4.21: Model of HandleConfiguration

Sender Responsibilities:

TestBATN server should manage the configuration data for each testcase party separately and ask the required data from the SUT administrators through TestBATN Control and Monitoring RIA.

Receiver Responsibilities:

RIA should present this information to the SUT administrator and ask for the related data fields conveniently. Upon user's answers, RIA should prepare the "HandleConfigurationResponse" message to be sent to the TestBATN server.

Repeatability:

No.

SUT administrator fills in all the configuration related information and clicks on the "Send" button. RIA prepares and sends a "HandleConfigurationResponse" message to the TestBATN server.

Interaction: HandleConfigurationResponse

Description:

This is the response of "HandleConfiguration" message including the filled in data by the SUT administrator.

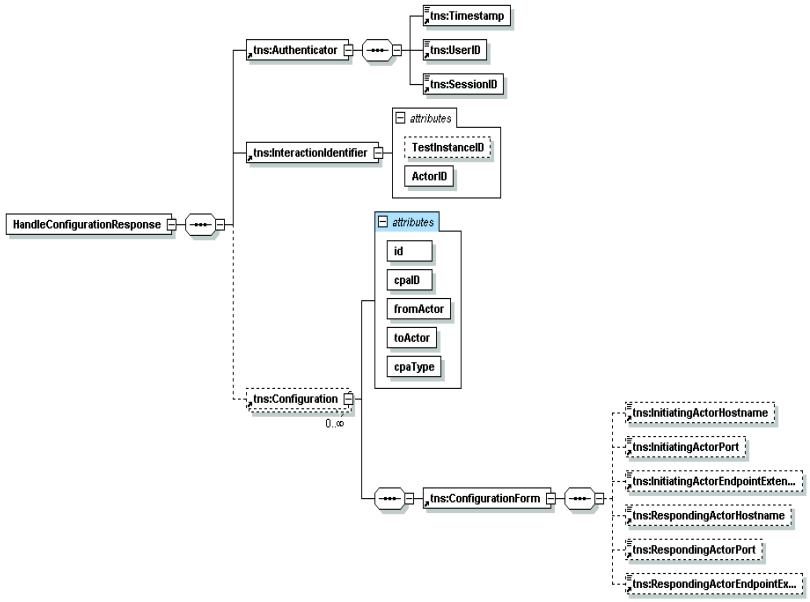

Figure 4.22: Model of HandleConfigurationResponse

Sender Responsibilities:

RIA must fill the corresponding fields of the message with the gathered information from the SUT administrator.

Receiver Responsibilities:

TestBATN server finalizes the configuration of the testcase after receiving this message from all participating parties of the testcase.

Repeatability:

No.

TestBATN server executes the testcase one step forward after collecting all configuration information from the related parties of the testcase. This step is the handling of the preliminary data if defined inside the testcase. If there is no defined preliminary data, the server sends the "SendTestSteps" message directly and the RIA behaves accordingly.

As shown in Figure 4.23, RIA maintains a list of the preliminary data sent within the "HandlePreliminaryTestData" interaction.

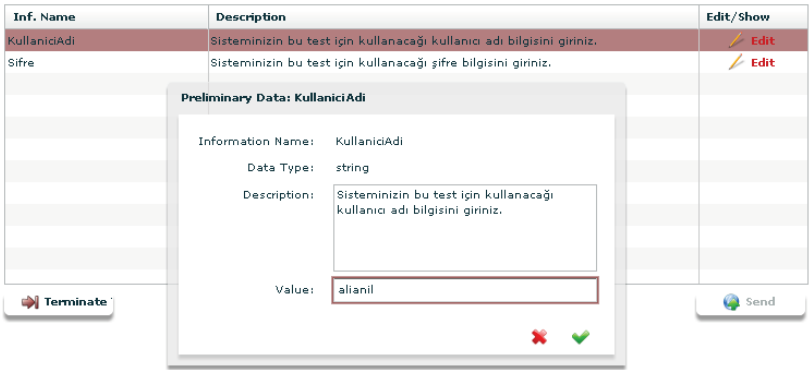

Figure 4.23: Handling Preliminary Data

Through the GUI, the user enters or views the preliminary test data information settled in the testcase definition. After the SUT administrator fills in the requested information through "HandlePreliminaryTestData" interaction, RIA sends this information to the TestBATN server with the "FilledInPreliminaryData" interaction.

Interaction: HandlePreliminaryTestData

Description:

This interaction exists to enable the semantic and functional testcases describe in Chapter 3.

Sender Responsibilities:

TestBATN server processes the preliminary data fields defined inside the testcase and creates the necessary structures.

Receiver Responsibilities:

RIA lists the preliminary data fields to the user and enables a friendly GUI to make the SUT administrator correctly fill the requested information. As a response to this message, RIA should prepare the corresponding "FilledInPreliminaryData" message and send to the server.

Repeatability:

No.

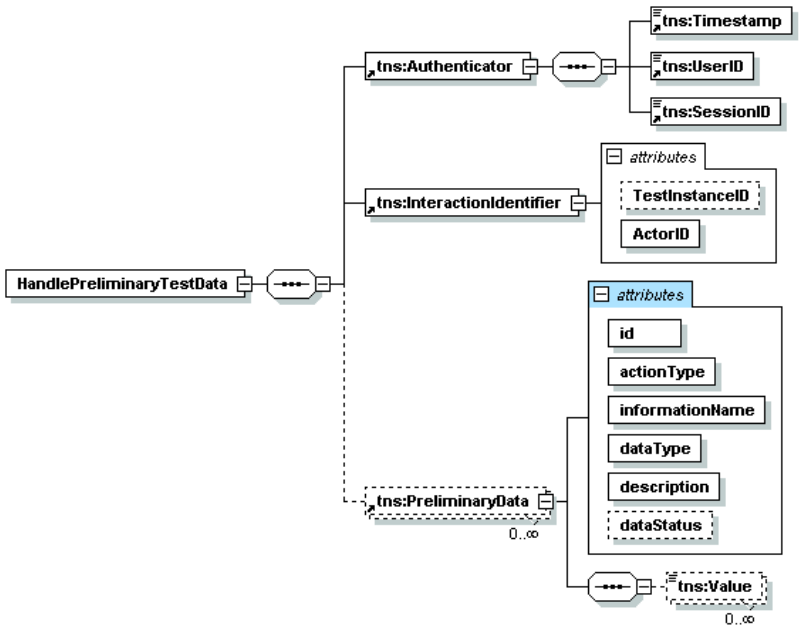

Figure 4.24: Model of HandlePreliminaryTestData

Interaction: FilledInPreliminaryData

Description:

This is the response of the "HandlePreliminaryTestData" interaction.

Sender Responsibilities:

RIA must correctly prepare and set the corresponding data fields inside message upon the SUT administrator inputs.

Receiver Responsibilities:

TestBATN server maintains a pool for the preliminary information collected from the parties of the testcase. If any "UpdatePreliminaryTestData" interaction is required to notify a user, the server should prepare the message and send it to the corresponding party/parties.

Repeatability:

No.

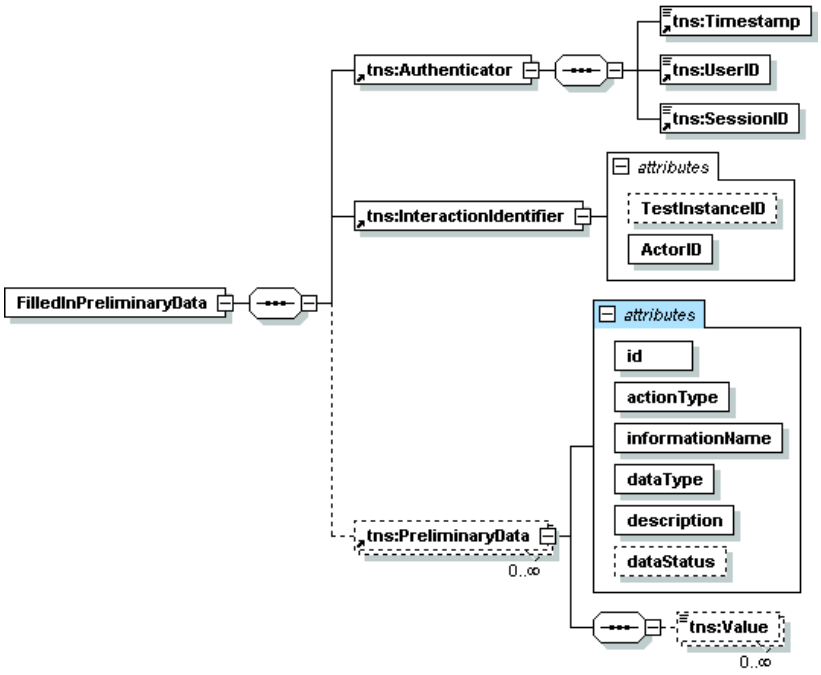

Figure 4.25: Model of FilledInPreliminaryData

In some testcase definitions, the test designer may create a test scenario in which the parties exchange information at the preliminary test data phase. For example, a one party may need information throughout the testcase execution which will be asked from another party. When the responder sends the information with "FilledInPreliminaryData" to the TestBATN server, the server checks whether any party is waiting for the retrieved preliminary data or not. If there exist a party, "UpdatePreliminaryTestData" interaction is used to notify the SUT administrator

for the updated value of the preliminary data field.

Interaction: UpdatePreliminaryTestData

Description:

Testcases defined among multiple parties may need exchange of preliminary information between the parties. When a party sends “FilledInPreliminaryData” to the server, it is sent to the waiting party, if exist, with this message.

Sender Responsibilities:

TestBATN server must wait for the completion of all “FilledInPreliminaryData” interaction and assignments of all the preliminary data variables.

Receiver Responsibilities:

RIA must present the user any update on the preliminary information upon the reception of this message.

Repeatability:

No.

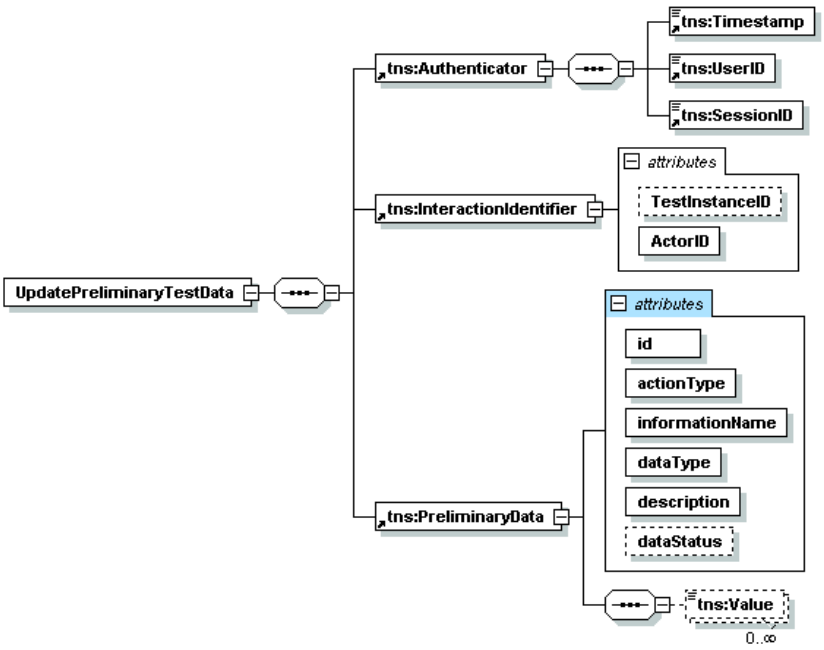

Figure 4.26: Model of UpdatePreliminaryTestData

As presented in Figure 4.17, after receiving all “FilledInPreliminaryData” interaction mes-

sages from the attending parties of the testcase, TestBATN server sends “UpdatePreliminary-TestData” to the parties, if there is any party waiting for. Afterwards, the server sends the details of the test steps of the testcase to the attending parties.

Figure 4.27 shows the snapshot of the GUI at after the “SendTestSteps” interaction is completed from the server to the RIA. SUT administrator can see the details of each test step, the test assertions which will be processed throughout the testcase execution and other details about the testcase.

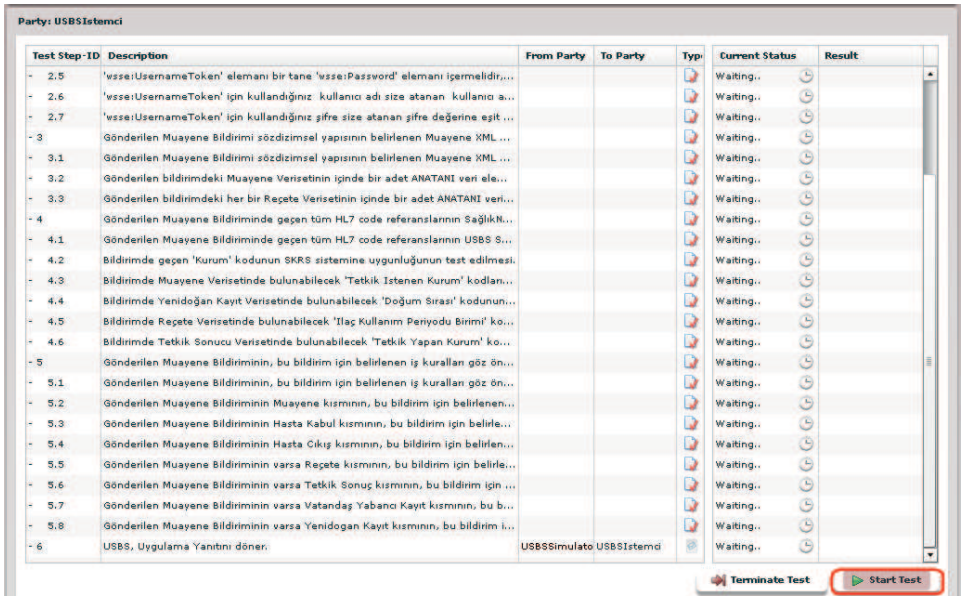

Figure 4.27: Test Steps View

Interaction: SendTestSteps

Description:

After the completion of the configuration management prior to the testcase execution, TestBATN server becomes ready to execute the scenario and sends this message to the RIA client to inform the SUT administrators about the details of the test steps.

Sender Responsibilities:

TestBATN server constructs the test steps through a convenient structure to be presented to the SUT administrator by the TestBATN Control and Monitoring RIA.

Receiver Responsibilities:

TestBATN Control and Monitoring RIA processes the received test steps and presents to the user as shown in Figure 4.27.

Repeatability:

No. For each testcase execution, this message is sent to each RIA client only once.

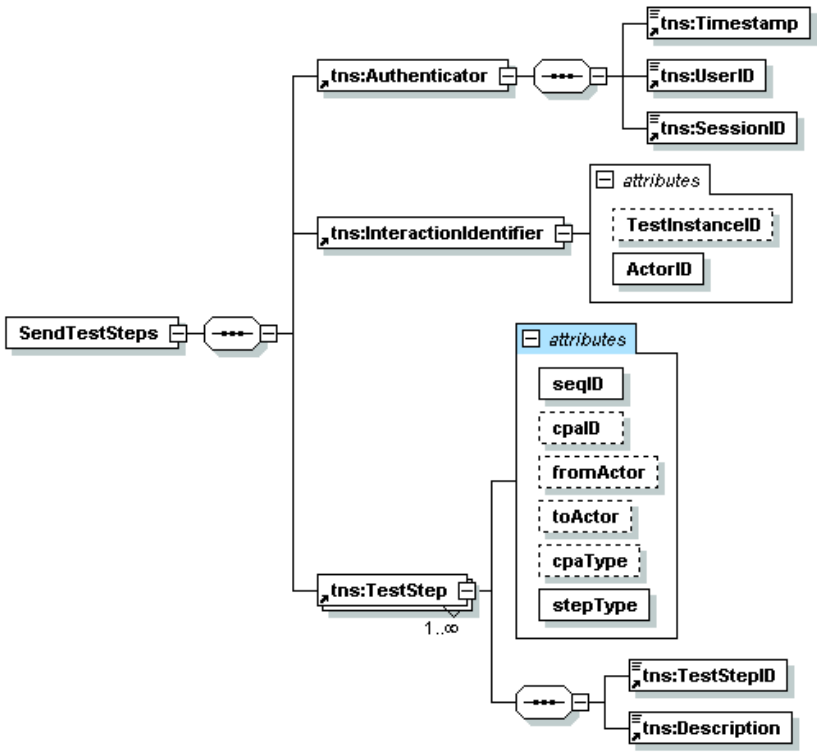

Figure 4.28: Model of SendTestSteps

TestBATN server sends the test steps to all parties of the testcase. Now, everthing is ready to start the execution of the testcase. To start this process, the parties of the testcase must all agree on the "StartTest" interaction. That is, all parties must send the "StartTest" message to the TestBATN server.

Interaction: StartTest

Description:

SUT administrator clicks on the "Start Test" button as shown in Figure 4.27. TestBATN Control and Monitoring RIA prepares and sends this message to the TestBATN server.

Sender Responsibilities:

RIA must send the message to the server upon user request by clicking on “Start Test” button.

Receiver Responsibilities:

TestBATN server waits for the “StartTest” message from all the parties of the test-case to start the execution of the testcase.

Repeatability:

No. A testcase can only be started once.

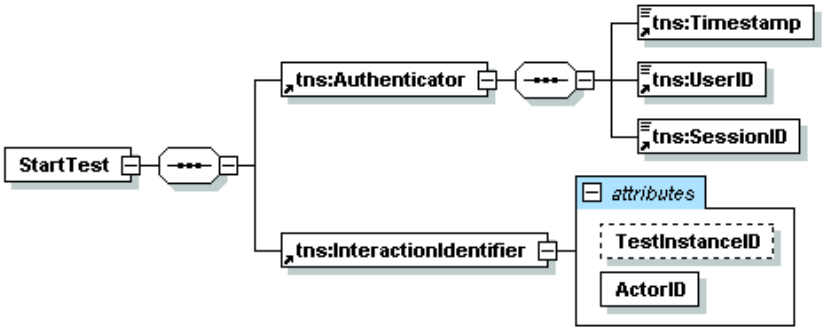

Figure 4.29: Model of StartTest

TestBATN server starts processing on the teststeps of the testcase one by one because all parties of the testcase agreed on the “StartTest”. Figure 4.30 shows the message choreography between the TestBATN server and TestBATN Control and Monitoring RIA just after the “StartTest” interaction until the testcase terminates totally with “ReleaseTestCasePorts” interaction.

When the server completes the execution of a teststep, it sends the result and the report of that teststep to the SUT administrators through TestBATN Control and Monitoring RIA. This notification is done through “UpdateTestStatus” interaction. RIA presents the changes on the status and results of the teststeps to the SUT administrator. Figure 4.31 shows the snapshot of the environment at the time of testcase execution. The server processes each teststep and sends the result to the RIA client immediately. Since the underlying Test Driving Protocol functions in high speed, SUT administrator sees the changes instantly on the GUI.

“UpdateTestStatus” interaction not only carries the result and status information about the

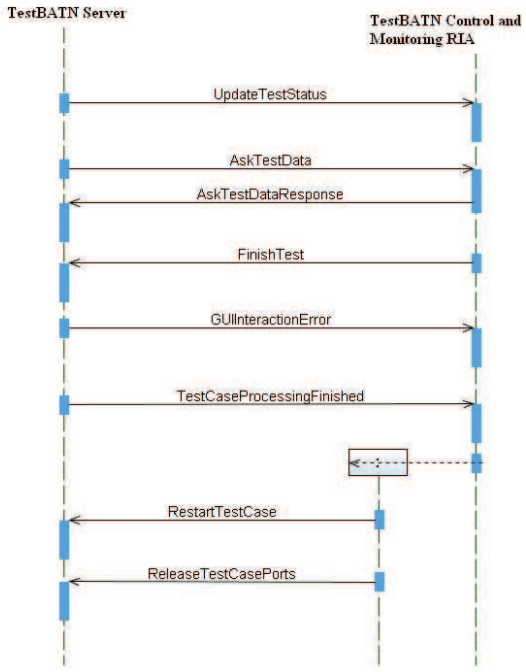

Figure 4.30: At the time of test execution

teststep, but informs the SUT administrator about the report of the execution of that teststep. As presented in Figure 4.32, TestBATN Control and Monitoring RIA presents the test step reports to the SUT administrator when clicked on the corresponding button appearing on the teststep status view section. For example, the report in the figure says that execution of the corresponding teststep has ended up with “FAIL” result. The validation adaptor which has created the report is the “xpathadaptor” adaptor. The reported error is about the Username - Token profile usage inside the SOAP header. The password field of the token does not match with the previously provided value through preliminary data handling.

Interaction: UpdateTestStatus

Description:

After the execution of each teststep, TestBATN server sends this message to the RIA clients to update the status, result and report of the corresponding teststep on the GUI of SUT administrators.

Sender Responsibilities:

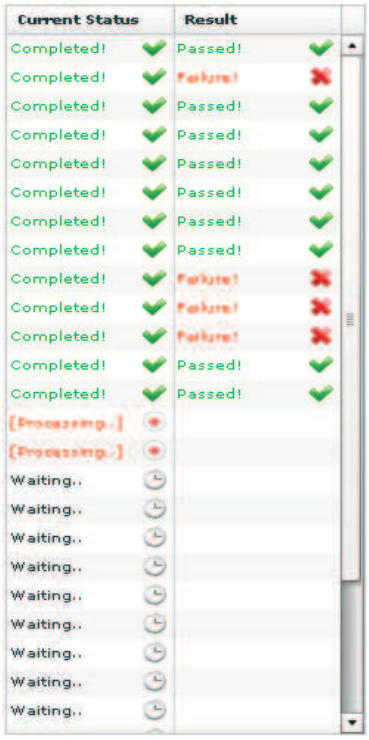

Figure 4.31: Teststep Status Changes

TestBATN server must monitor the execution of the teststeps and send this message to all attending testcase parties when the processing of each teststep ends.

Receiver Responsibilities:

TestBATN Control and Monitoring RIA is responsible for updating the corresponding view components to notify the SUT administrator about the changes.

Repeatability:

Yes. For each update on each teststep, this message is created and sent to the RIA clients.

Semantic and functional testing capabilities of TestBATN include a mechanism to ask some questions to the SUT administrator at the time of execution of the testcase. These questions are asked through the “AskTestData” interaction and the answers of the SUT administrator are carried back to the server through “AskTestDataResponse” interaction.

Interaction: AskTestData

Description:

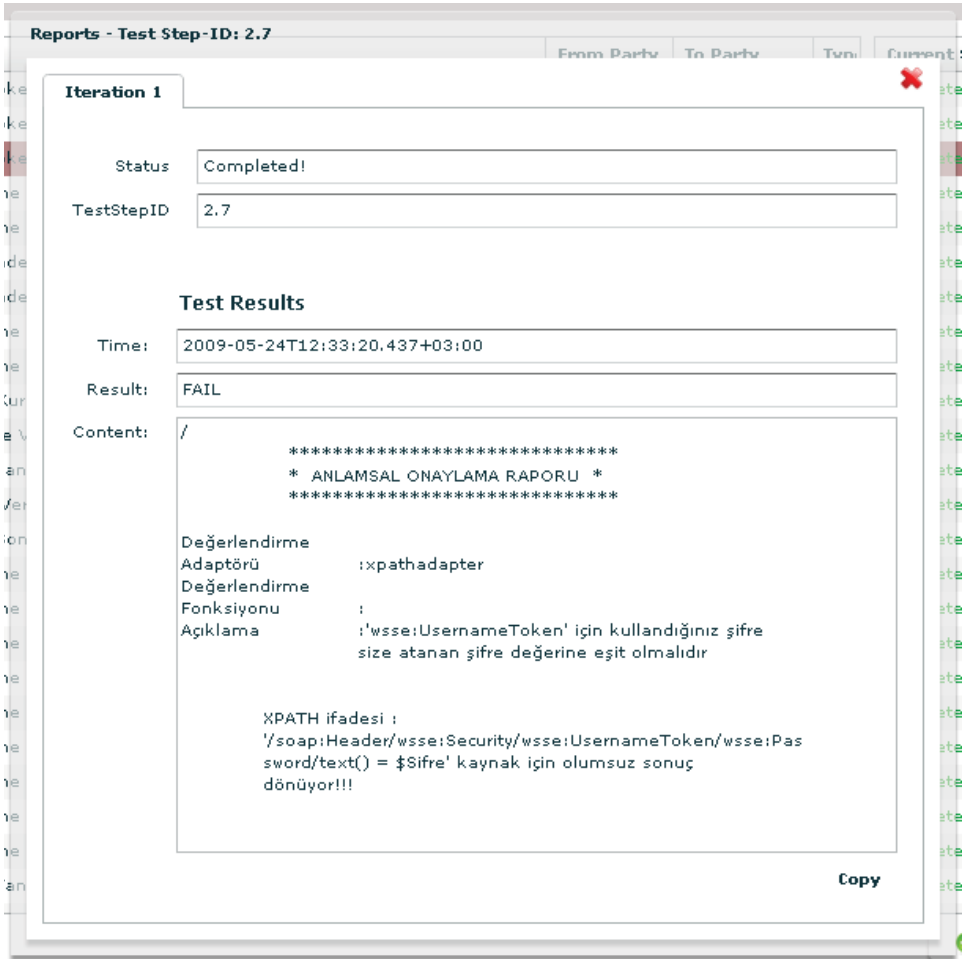

Figure 4.32: Teststep Report

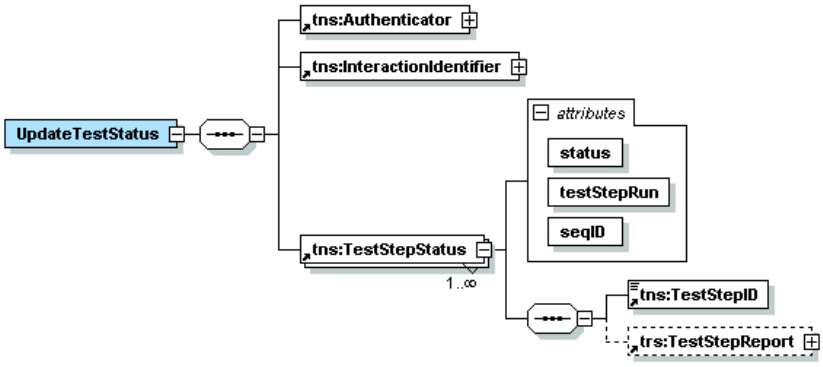

Figure 4.33: Model of UpdateTestStatus - I

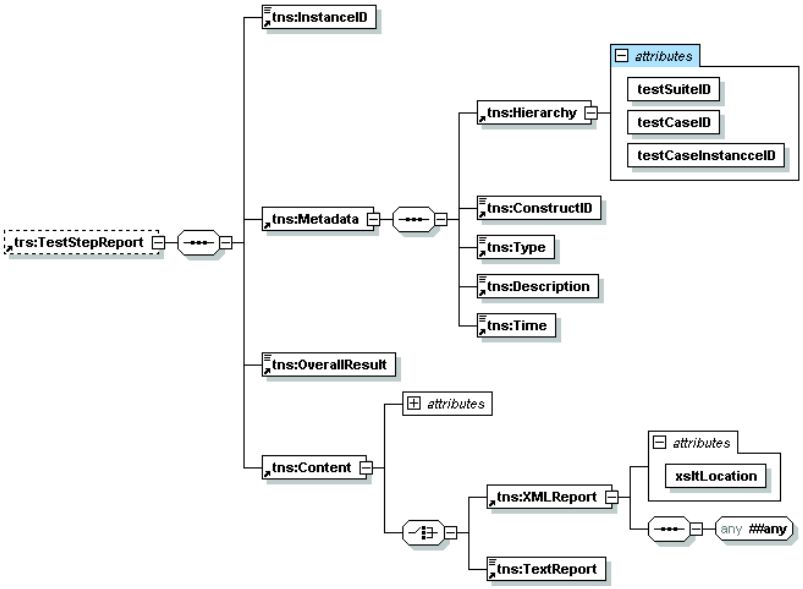

Figure 4.34: Model of UpdateTestStatus - II

If defined in the testcase, this message is created and sent from the TestBATN server to the RIA to ask the questions indicated in the testcase to the SUT administrator.

Sender Responsibilities:

TestBATN server must create and send this message to the corresponding user's RIA when encounters an "AskTestData" construct of Test Description Language.

Receiver Responsibilities:

TestBATN Control and Monitoring RIA pops up a window to ask the questions and gather the responses from the SUT administrator.

Repeatability:

Yes. Throughout the testcase definition, for each "AskTestData" construct of TDL, this message is sent to the corresponding RIA client.

Interaction: AskTestDataResponse

Description:

This is the reponse of "AskTestData" returned from the RIA to the TestBATN server.

Sender Responsibilities:

TestBATN Control and Monitoring RIA should correctly set the fields inside the message according to the SUT administrator's answers.

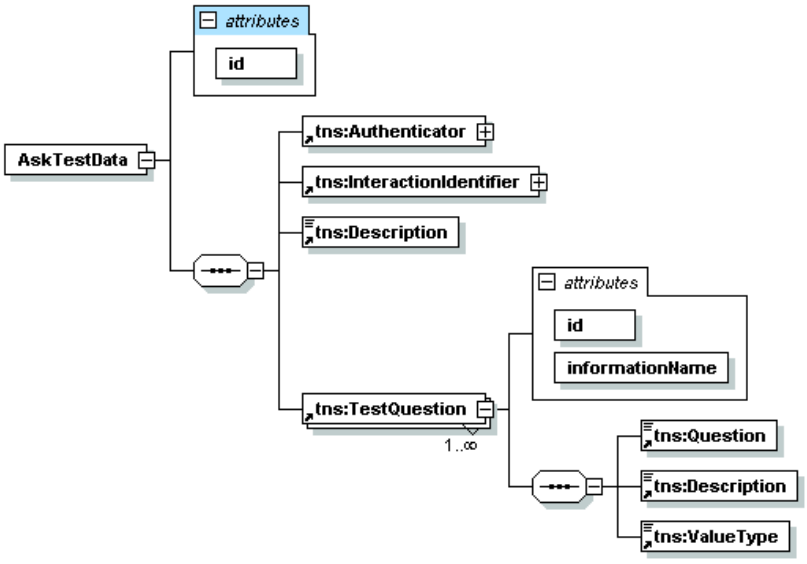

Figure 4.35: Model of AskTestData

Receiver Responsibilities:

TestBATN server runs the asserted operations on the responses of the SUT administrator as settled through the testcase definition.

Repeatability:

No. This message can only be sent as a response of “AskTestData” interaction. If “AskTestData” repeats, “AskTestDataResponse” message also repeats as a response.

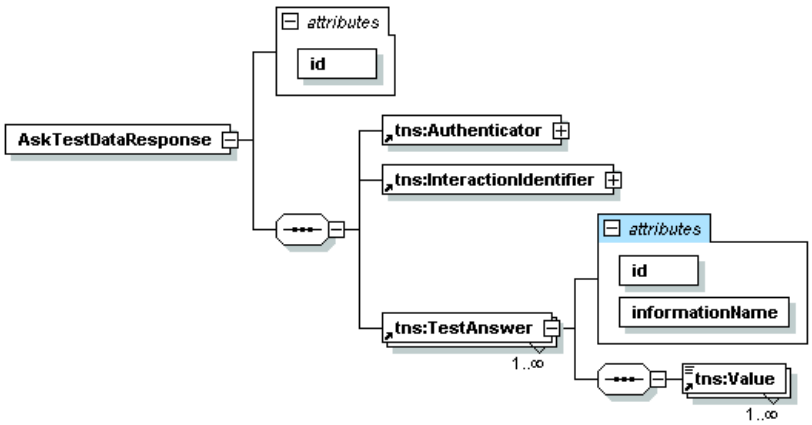

Figure 4.36: Model of AskTestDataResponse

SUT administrator has a chance to stop the execution of the testcase and quit to the main page of the TestBATN Control and Monitoring GUI at any time. This operation is handled through the “FinishTest” interaction which is sent from the RIA to the TestBATN server when the SUT administrator clicks on the “Terminate Test” button.

Interaction: FinishTest

Description:

This message indicates that SUT administrator wants to quit the execution of the testcase.

Sender Responsibilities:

RIA must send this message to the TestBATN server when the SUT administrator clicks on “Terminate Test” or “Log out” button.

Receiver Responsibilities:

TestBATN server processes this message and informs the other parties of the testcase about the termination of the execution through the “TestCaseProcessingFinished” interaction. If any of the parties of the testcase terminates the testcase, it terminates for all of the parties.

Repeatability:

No.

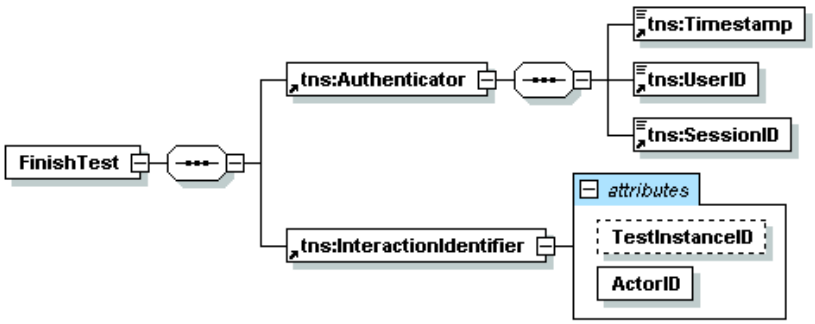

Figure 4.37: Model of FinishTest

Interaction: TestCaseProcessingFinished

Description:

This message is sent from the TestBATN server to the RIA clients who are executing the testcase in question at that time to to inform them about the termination of the testcase.

Sender Responsibilities:

TestBATN server sends this message to the testcase parties when the tetscase finishes normally by executing all teststeps or one of the parties terminates the execution of the testcase in the interoperability tests.

Receiver Responsibilities:

RIA notifies the user by popping up an information window about the termination and the final report of the testcase.

Repeatability:

No. This message is sent only once to the RIA client to indicate the termination of the execution.

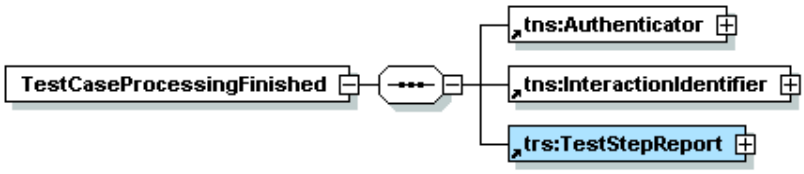

Figure 4.38: Model of TestCaseProcessingFinished

After the execution of a testcase terminates, the SUT administrators are informed with this event and the TestBATN Control and Monitoring RIA behaves accordingly by updating the views of the user as seen in Figure 4.39. When the user clicks on the “Restart” button, RIA sends the “RestartTestCase” message to the TestBATN server to restart the execution of the testcase with the already configured parameters. The GUI returns to the preliminary data handling state to enable the SUT administrator to change the entered values. If the SUT administrator wants to go back to main page to select another testcase or do any other operation, “Go Back” button is clicked. With this user event, RIA sends the “ReleaseTestCasePorts” message to the TestBATN server.

Interaction: RestartTestCase

Description:

Figure 4.39: Restart of a Testcase

With this message, RIA realizes the re-execution of the lastly executed testcase with the same configuration parameters.

Sender Responsibilities:

TestBATN Control and Monitoring RIA must give the choice of restarting the test-case to the SUT administrator as shown in 4.39.

Receiver Responsibilities:

TestBATN server must keep the configuration information of the testcase even if it is terminated until the “ReleaseTestCasePorts” message is received from the RIA.

Repeatability:

Yes. A testcase execution can be repeated with the same configuration parameters as much as requested by the SUT administrator.

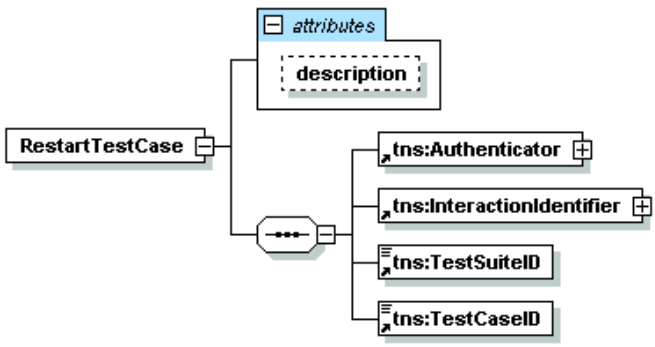

Figure 4.40: Model of RestartTestCase

Interaction: ReleaseTestCasePorts

Description:

When the SUT administrator does not want to re-execute the testcase, he/she clicks on “Go Back” button as shown in Figure 4.39. This message, then, is sent to the TestBATN server to reset the configuration parameters of the lastly executed test-case by the user to free the occupied ports and other intermediates.

Sender Responsibilities:

TestBATN Control and Monitoring RIA must provide the choice of going back to the main page of the environment to the SUT administrator.

Receiver Responsibilities:

TestBATN server deletes the configuration information of the testcase and releases the occupied ports.

Repeatability:

No.

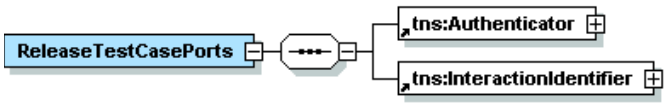

Figure 4.41: Model of ReleaseTestCasePorts

During the interaction between TestBATN server and TestBATN Control and Monitoring Environment, if any internal error occurs in the TestBATN server it is reported to the RIA clients through the “GUIInteractionError” interaction.

Interaction: GUIInteractionError

Description:

This message is sent from the TestBATN server to RIA clients of all parties of the testcase in case of an internal error.

Sender Responsibilities:

Errors occurred inside the TestBATN server are caught and sent to the RIA clients with convenient reports.

Receiver Responsibilities:

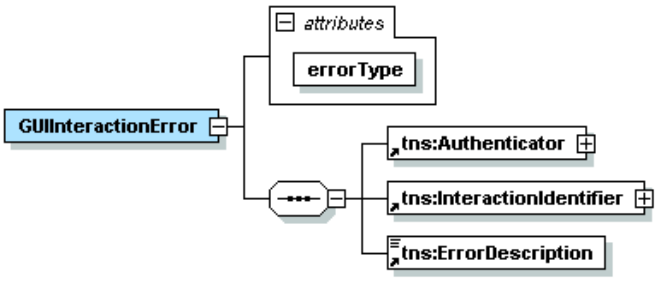

Figure 4.42: Model of GUIInteractionError

TestBATN Control and Monitoring RIA is responsible to handle the presentation issues when this message is received. The error report is shown to the user and the state of the view is arranged according to the termination issues.

Repeatability:

Yes. At each time any error occurs, this message is sent from the server to the clients.

CHAPTER 5

RELATED WORK

While e-business scenarios are widely adopted by the users in industry, governments and the public sector, it is still cumbersome for them to reach interoperability of eBusiness solutions and to achieve conformance with standards specifications. Therefore, the need for advanced testing methodologies and practices which cover relevant set of standards is increasing continuously.

There are a number of research, development and even standardization efforts on testing methodologies, test tools, languages and frameworks.

Testing and Test Control Notation (TTCN) [72] is one of the first and most successful work on test automation published by the European Telecommunications Standards Institute (ETSI) [78]. The latest version, TTCN-3 is a computationally complete and internationally standardized programming language for expressing test cases for conformance and interoperability testing in the telecommunication domain.

Telecommunication networks use many heterogenic protocols between multiple system nodes, with dedicated configuration and application data. Ensuring the interoperability of the components within such systems, across different vendor platforms and countries require testing the network components accurately. For this purpose, ETSI has developed methodologies for:

- conformance testing for checking the compliance of isolated components of the telecommunication networks, to the protocol standards
- interoperability testing to verify the ability of network components from different vendors to operate in real conditions

ETSI provides a number of publicly available TTCN-3 test suites such as Digital Mobile Radio (DMR), digital Public Mobile Radio (dDMR), Dynamic Host Configuration Protocol (DHCPv6), IPv6, IP Multimedia Subsystem (IMS), Voice Over IP with the Session Initiation Protocol (SIP), and the upcoming WiMax test suite.

Since the telecommunications domain is concerned with the low level issues by nature, TTCN test suites and testing methodologies also focuses on the low level details of the communication. As a result, it does not fully represent the business processes that mostly the concern of the high level standards such as the ones used in NHIS. Furthermore, TTCN does not provide any interactive, human-driven test framework. The test suites are static and for each test scenario and for each client to be tested, specialized code fragments are needed to be developed prior to the test scenario executions.

OASIS IIC ebXML Test Framework [79] is a test framework enabling conformance and interoperability testing for ebXML specifications. The framework includes an architecture design based on components that can be combined and distributed in different ways, to accommodate different test harnesses. It also includes an extensible test scripting language for coding test suites in an executable way. In other words, it describes a test bed architecture and its software components as well as how these can be combined to create a test harness for various types of testing [69].

OASIS IIC ebXML Test Framework is totally based on ebXML messaging; therefore, it focuses on its Message Service Handler (MSH) implementations instead of application testing. Although the specification says that the framework has support for external plug-ins, it would not be possible to test NHIS services through this framework not only due to the need of those external plugins which enable the processing of HL7 based messages through the Web services profile but also the lack of an interactive, human-driven test control and monitoring environment.

OASIS Event Driven Test Scripting Language (eTSL) [80], which is currently a work in progress, improves OASIS IIC ebXML Test Framework by addressing the different layers of the interoperability stack, namely, the messaging infrastructure, the messaging choreographies and the business document standards. eTSL is a model and language for eBusiness / eGovernment test suites, with a particular focus on communication events, versatile usage for the Quality Assurance testing phase as well as the monitoring of deployed systems, and

extensible design to leverage specialized validation processors as well as XML tools such as XPath [15], XSLT [18] and XQuery [81].

NIST B2B Testbed [82] project includes a sample implementation of OASIS IIC ebXML Test Framework v1.1. NIST B2B Testbed is initiated by the National Institute of Standards and Technology [83] to develop software tools which can be used to test current B2B standards or technologies and enhance the capabilities for on-demand demonstration of conformance and interoperability [84].

TestBATN is a superior testing framework than the OASIS and ETSI frameworks with the added value of the TestBATN Control and Monitoring Environment, presented in Chapter 4, which enables the human-driven test cases. Indeed, in the TestBATN framework an extended version of the event notion of eTSL is provided which helps capture the requirements of the test scenarios encountered in the testing of NHIS. These requirements include interaction with the users of a System Under Test (SUT), using external evaluation services, and allowing human validation during test execution. Furthermore, preliminary test steps such as configuration management and test data initialization have been included in the test execution of TestBATN. This functionality is critical in the sense that it allows the test framework to adapt itself to the requirements of the applications under test, rather than the opposite. Finally, the TestBATN framework provides an extensible and layered platform for design, execution and life cycle management of the test scenarios to handle the dynamically evolving testing requirements through the TestBATN Control and Monitoring Environment.

There are also some eHealth specific tools for the conformance testing. HL7 Message Maker [85] is an initiative administered by HL7 together with NIST [83] to automatically and dynamically generate test messages for eHealth applications. The data used to populate the messages is drawn from a number of sources including the NIST developed database of HL7 data items, HL7 tables, user tables, and external tables. This initiative focuses on the document content layer within the interoperability stack. Testing of NHIS requires the test assertions through the whole layers of the interoperability stack since the Integration Profile includes several standards and specifications targeting to the different layers of the stack.

IHE provides a detailed implementation and testing process to promote the adoption of standards-based interoperability by vendors and users of healthcare information systems. The process is named as Connect-a-thon, a weeklong interoperability-testing event. Managing

a Connect-a-thon is a human labor intensive task. Most of the messaging is conducted by the humans such as checking message contents or application behaviours. Connect-a-thons require the physical aggregation of the software components and the administrators of those softwares for a week through a set of workshops. However, the TestBATN framework provides the TestBATN Control and Monitoring Environment for the automated execution of the test cases. Therefore, testing of NHIS would be a very complex operation through several Connect-a-thon organizations.

CHAPTER 6

CONCLUSION AND FUTURE WORK

The final outcome of this thesis enabled the conformance and interoperability testing of the applications developed within the National Health Information System (NHIS) of Turkey. Nearly all of the Family Medicine Information System (FMIS) and Hospital Information System (HIS) vendors somehow tested their implementations through TestBATN with the testcases developed in the scope of this thesis.

Specifications, standards and the Implementation/Integration/Interoperability Profile of the Ministry of Health (MoH), Turkey, are analyzed and the requirements to enable the testing of the system are put on the map. According to the identified testing requirements, a testing methodology is designated which follows a test procedure ranging from the basic conformance tests to the complex semantic and functional tests. In respect to the designated testing methodology, several testcases; such as basic conformance testcases, customized conformance testcases, interoperability testcases, and testcases for the query, update and delete methods of the NHIS services and a number of semantic testcases for the NHIS services in question are developed and registered to the TestBATN framework.

Considering the testing requirements of NHIS, TestBATN Control and Monitoring Environment is designed and implemented within the scope of this thesis. The environment realized a human-driven testing infrastructure. Developing a Rich Internet Application as the Graphical User Interface to the FMIS/HIS administrators and implementing a Test Driving Protocol provided a comprehensive, user-friendly, simple and high speed control and monitoring environment for the TestBATN users.

As mentioned earlier, NHIS servers located at the MoH premises has been accepting “Transmission Schema” instances since November, 2008. As of January 15, 2009, all healthcare

institutes have to send the actual, real patient EHR data to NHIS servers with the announcement of governmental regulations.

Currently, nearly all of the state hospitals of Turkey are able to successfully send the daily EHR messages, which are the “Transmission Schema” instances to the NHIS servers. University hospitals and primary care institutions are continuously boosting up their integration capabilities with Sağlık-Net.

Up to 70 companies with nearly 340 users have been testing their systems through the developed testcases through “http://www.srdc.com.tr/testbatn” since June, 2008 as mentioned earlier in the thesis. Since then, approximately 20.000 testcase executions are recorded. Not only the developed testcases through TestBATN was availbale online, but also two well-attended integration workshops were organized in Turkey. The first one was in Çeşme, İzmir during June, 29 and July, 5, 2008. This workshop has provided various opportunities to nearly 130 FMIS/HIS developers from 50 vendor companies about testing their applications successfully with the developed testcases. The second workshop was held in Ankara during December 19 - 21, 2008. 133 attendees from 64 vendor companies were there to test their implementations.

Together with the continuous online use of the developed testcases through TestBATN and the integration workshops organized by the MoH, in which the developed testcases through TestBATN were also used, increased the integration capabilities of the FMISs and HISs developed for NHIS. The feedbacks coming from the vendor companies, the statistics of the ongoing integration and the representatives of the Ministry of Health confirm the contribution of the TestBATN framework and the comprehensive testcases to the integration skills of the client applications. Furthermore, the testcases and the user-friendly, high speed TestBATN Control and Monitoring Environment increased the awareness of the software companies, especially in the health sector of Turkey, in the international standards, their use in real life etc. This process has also revealed the importance of the adoption of international standards within such integration efforts, like NHIS.

The work presented in this thesis is supported by the TÜBİTAK TEYDEB Project No: 7070191 in addition by the Ministry of Health, Turkey.

NHIS is collecting the Electronic Health Records (EHR) to the central servers located at the MoH premises, currently. The use of this data is for statistical purposes for now. However,

the ongoing process is going towards the integration of the hospitals within each other. That is, in the near future, the goal is to enable the sharing of the EHRs among the hospitals instead of only collecting to the center. This process requires further profiles and business models and more intensive interoperability capabilities. TestBATN can easily be used in the testing of these interoperability capabilities of the FMISs and HISs. New testcases for the interoperability scenarios can be developed and online use of the TestBATN framework can enable the joint test executions without requiring the software applications' coming together "physically".

Furthermore, TestBATN can be used to certify the client applications within NHIS to assure their conformance and interoperability skills and enable a more intact and accurate integration.

REFERENCES

[1] IEEE Dictionary, Institute of Electrical and Electronics Engineers. IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries, New York, 1990.

[2] Brown and Reynolds, "Strategy for production and maintenance of standards for interoperability within and between service departments and other healthcare domains", CEN/TC 251 Health Informatics, CEN/TC 251/N00-047.

[3] ETSI EG 202 237, Methods For Testing and Specification. http://portal.etsi.org/mbs/-Referenced%20Documents/eg_202_237.pdf, last visited on June 2009.

[4] e-Transformation in Health. Department of Information Processing, Ministry of Health, Turkey. http://www.saglik.gov.tr/EN/Tempdosyalar/533__e-transformationinhealth_07.pdf, last visited on June 2009.

[5] Ulusal Sağlık Bilgi Sistemi/Sağlık-NET Entegrasyonu ile İlgili Genelge 2008/18. http://-www.saglik.gov.tr/TR/MevzuatGoster.aspx?F6E10F889 2433CFF1A9547B61DAFFE-2A56515916B329A1F1, last visited on June 2009.

[6] Health Level 7. http://www.hl7.org/, last visited on June 2009.

[7] American National Standards Institute (ANSI). http://www.ansi.org/, last visited on June 2009.

[8] Clinical Document Architecture (CDA), Release 2. http://www.hl7.org/v3ballot/html/-infrastructure/cda/cda.htm, last visited on June 2009.

[9] Extensible Markup Language (XML). http://www.w3.org/XML/, last visited on June 2009.

[10] Mustafa Yuksel, Sharing Electronic Healthcare Records Across Country Borders, Master's thesis, Middle East Technical University, Department of Computer Engineering, Ankara, Turkey, 2008.

[11] Transmission Control Protocol (TCP). http://www.faqs.org/rfcs/rfc793.html, last visited on June 2009.

[12] Namli T., Aluc G., Sinaci A., Kose I., Akpinar N., Gurel M., Arslan Y., Ozer H., Yurt N., Kirici S., Sabur E., Ozcam A., Dogac A. Testing the Conformance and Interoperability of NHIS to Turkey's HL7 Profile 9th International HL7 Interoperability Conference (IHIC) 2008, Crete, Greece, October, 2008, pp. 63-68.

[13] XML Schema Definition (XSD). http://www.w3.org/XML/Schema, last visited on June 2009.

[14] World Wide Web Consortium. http://www.w3.org/, last visited on June 2009.

[15] XML Path Language (XPath). http://www.w3.org/TR/xpath, last visited on June 2009.

[16] The Schematron. http://xml.ascc.net/resource/schematron/, last visited on June 2009.

[17] The Schematron. http://www.schematron.com, last visited on June 2009.

[18] XSL Transformations (XSLT). http://www.w3.org/TR/xslt, last visited on June 2009.

[19] International Organization for Standardization. http://www.iso.org, last visited on June 2009.

[20] HyperText Transfer Protocol (HTTP). http://www.w3.org/Protocols/, last visited on June 2009.

[21] HyperText Markup Language (HTML). http://www.w3.org/TR/REC-html40/, last visited on June 2009.

[22] PHP. http://www.php.net/, last visited on June 2009.

[23] ASP.NET White Papers. http://www.asp.net/Learn/whitepapers/, last visited on June 2009.

[24] Java Server Pages Technology - White Paper. http://java.sun.com/products/jsp/-whitepaper.html, last visited on June 2009.

[25] ColdFusion Datasheets and White Papers. http://www.adobe.com/products/coldfusion/-whitepapers/, last visited on June 2009.

[26] JavaScript on Wikipedia. http://en.wikipedia.org/wiki/JavaScript, last visited on June 2009.

[27] ActionScript on Wikipedia. http://en.wikipedia.org/wiki/ActionScript, last visited on June 2009.

[28] Farrell, Jason. “Rich Internet Applications: The Next Stage of Application Development”. Online. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&-arnumber=4283806&isnumber=4283720

[29] Adobe Flex. http://www.adobe.com/devnet/flex/, last visited on June 2009.

[30] Adobe Flash. http://www.adobe.com/products/flash/, last visited on June 2009.

[31] Java Web Start. http://java.sun.com/javase/technologies/desktop/javawebstart/index.jsp, last visited on June 2009.

[32] Java Network Launch Protocol (JNLP). http://jcp.org/en/jsr/detail?id=056, last visited on June 2009.

[33] JavaFX. http://javafx.com/, last visited on June 2009.

[34] Ajax. http://www.ajax.org/, last visited on June 2009.

[35] Cascading Style Sheets (CSS). http://www.w3.org/Style/CSS/, last visited on June 2009.

[36] Document Object Model (DOM). http://www.w3.org/DOM/, last visited on June 2009.

[37] XML User Interface Language (XUL). http://www.mozilla.org/projects/xul/, last visited on June 2009.

[38] Adobe Flash Player. http://www.adobe.com/products/flashplayer/, last visited on June 2009.

[39] SWF. http://www.adobe.com/devnet/swf/, last visited on June 2009.

[40] ECMAScript. http://www.ecma-international.org/publications/standards/Ecma-357.htm, last visited on June 2009.

[41] Simple Object Access Protocol (SOAP). http://www.w3.org/TR/soap/, last visited on June 2009.

[42] Cairngorm White Paper. http://www.adobe.com/devnet/flex/articles/-introducing_cairngorm/introducing_cairngorm.pdf, last visited on June 2009.

[43] Cairngorm Micro-architecture Diagram. http://www.cairngormdocs.org/-cairngormDiagram/cairngorm2_rpc.swf, last visited on June 2009.

[44] H. Zimmermann. OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection. IEEE Transactions on Communications. COM-28: 425-432, 1980.

[45] HL7 Message Development Framework, Version 3.3, December 1999. http://-www.hl7.org/Library/MDF99/MDF99doc.zip, last visited on June 2009.

[46] HL7 Reference Information Model. http://www.hl7.org/v3ballot/html/infrastructure/-rim/rim.htm, last visited on June 2009.

[47] The National Health Data Dictionary (NHDD) of Turkey. http://-www.sagliknet.saglik.gov.tr/portal_pages/notlogin/bilisimciler/docs/-usvs_sozluk_1.1.rar, last visited on June 2009.

[48] HL7 Hierarchical Message Description (HMD). http://www.hl7.org/v3ballot/html/help/-v3guide/v3guide.htm#v3ghmd, last visited on June 2009.

[49] HL7 Vocabulary. http://www.hl7.org/v3ballot/html/infrastructure/vocabulary/-vocabulary.htm, last visited on June 2009.

[50] HL7 Data Types - Abstract Specification. http://www.hl7.org/v3ballot/html/-infrastructure/datatypes/datatypes.htm, last visited on June 2009.

[51] HL7 Refinement, Constraint and Localization, Release 2. http://www.hl7.org/v3ballot/-html/infrastructure/conformance/conformance.htm, last visited on June 2009.

[52] HL7 Version 3 Standard: Transport Specifications Overview. http://www.hl7.org/-v3ballot2008may/html/infrastructure/transport/transport-intro.htm, last visited on June 2009.

[53] HL7 Version 3 Standard: Transport Specification - ebXML, Release 2. http://-www.hl7.org/v3ballot/html/infrastructure/transport/transport-ebxml.htm, last visited on June 2009.

[54] HL7 Version 3 Standard: Transport Specification - Web Services Profile, Release 2. http://www.hl7.org/v3ballot/html/infrastructure/transport/transport-wsprofiles.htm, last visited on June 2009.

[55] Transport Specification: Minimal Lower Layer Message Transport Protocol (MLLP), Release 2. http://www.hl7.org/v3ballot/html/infrastructure/transport/transport-mllp.htm, last visited on June 2009.

[56] Web Services Description Language (WSDL). http://www.w3.org/TR/wsdl, last visited on June 2009.

[57] List of Web service specifications on Wikipedia. http://en.wikipedia.org/wiki/-List_of_Web_service_specifications, last visited on June 2009.

[58] Kabak Y., Dogac A., Kose I., Akpinar N., Gurel M., Arslan Y., Ozer H., Yurt N., Ozcam A., Kirici S., Yuksel M., Sabur E. The Use of HL7 CDA in the National Health Information System (NHIS) of Turkey. 9th International HL7 Interoperability Conference (IHIC) 2008, Crete, Greece, October 2008.

[59] Kose I., Akpinar N., Gurel M., Arslan Y., Ozer H., Yurt N., Kabak Y., Yuksel M., Dogac A. Turkey's National Health Information System (NHIS). In the Proceedings of the eChallanges Conference, Stockholm, October 2008.

[60] The Health Coding Reference Server 2. http://sbu.saglik.gov.tr/SKRS2%5FListesi/, last visited on June 2009.

[61] International Statistical Classification of Diseases and Related Health Problems, 10th Revision (ICD-10), Second Edition. Technical Report, World Health Organization, Geneva, Switzerland. http://www.who.int/whosis/icd10/, last visited on June 2009.

[62] Information technology – Metadata registries (MDR) – Part 4: Formulation of data definitions, (ISO/IEC 11179-4).

[63] Doctor Data Bank. http://sbu.saglik.gov.tr/drBank/, last visited on June 2009.

[64] Web Services Security: SOAP Message Security 1.1 (WS-Security 2004). OASIS Standard Specification. http://www.oasis-open.org/committees/download.php/16790/-wss-v1.1-spec-os-SOAPMessageSecurity.pdf, last visited on June 2009.

[65] Generation and registration of Universally Unique Identifiers (UUIDs) and their use as ASN.1 Object Identifier components. Open Systems Interconnection, International Telecommunication Union. http://www.itu.int/ITU-T/studygroups/com17/oid/-X.667-E.pdf, last visited on June 2009.

[66] MERNIS, Central Demographics Management System. http://www.nvi.gov.tr/-Hakkimizda/Projeler,Mernis_Genel.html?pageindex=4, last visited on June 2009.

[67] Saglik-Net Business Rules. Ministry of Health. http://www.sagliknet.saglik.gov.tr/-portal_pages/notlogin/bilisimciler/docs/SaglikNET_is_Kurallari_BRMS.pdf, last visited on June 2009.

[68] Scientific and Technological Research Council of Turkey (TÜBİTAK), TEYDEB Project No: 7070191.

[69] Namli T., Aluc G., Dogac A. An Interoperability Test Framework for HL7 based Systems IEEE Transactions on Information Technology in Biomedicine Vol.13, No.3, May 2009, pp. 389-399.

[70] ETSI Plugtests. http://www.etsi.org/WebSite/OurServices/plugtests/home.aspx, last visited on June 2009.

[71] Connectathon Fact Sheet. http://www.ihe.net/Connectathon/upload/-NA_2008_Connectathon_Fact_Sheet_1.pdf, last visited on June 2009.

[72] ETSI TTCN-3. http://www.ttcn-3.org/StandardSuite.htm, last visited on June 2009.

[73] Simple Mail Transfer Protocol (SMTP). http://www.ietf.org/rfc/rfc0821.txt, last visited on June 2009.

[74] OASIS ebXML Message Service Specification (ebMS). http://www.oasis-open.org/-committees/ebxml-msg/documents/ebMS_v2_0.pdf, last visited on June 2009.

[75] Minimum Health Data Sets (MHDS). http://www.sagliknet.saglik.gov.tr/portal_pages/-notlogin/bilisimciler/docs/msvs_semalari.rar, last visited on June 2009.

[76] The ebXML Test Framework and the Challenges of B2B Testing. http://-ebxmltesting.nist.gov/xmleurope/xmleurope.html, last visited on June 2009.

[77] Integrating the Healthcare Enterprise (IHE). http://www.ihe.net/, last visited on June 2009.

[78] Europen Telecommunications Standard Institute (ETSI). http://www.etsi.net/WebSite/-homepage.aspx, last visited on June 2009.

[79] OASIS ebXML Test Framework v1.0. http://www.oasis-open.org/committees/-download.php/10896/IIC_ebXMLTestFramework_v1.1_10_11_04_final_draft.zip, last visited on June 2009.

[80] Event-driven Test Scripting Language (eTSL), OASIS ebXML Implementation Interoperability and Conformance (IIC) TC, Working Draft 0.85. http://www.oasis-open.org/-committees/download.php/26036/eTSL-draft-085.pdf, last visited on June 2009.

[81] XML Query Language (XQuery). http://www.w3.org/TR/xquery/, last visited on June 2009.

[82] NIST Manufacturing Business to Business (B2B) Interoperability Testbed. http://-www.mel.nist.gov/msid/b2btestbed/, last visited on June 2009.

[83] National Institute of Standards and Technology (NIST). http://www.nist.gov/, last visited on June 2009.

[84] B. Kulvatunyou, N. Ivezic, M. Martin, A. T. Jones, A BusinesstoBusiness Interoperability Testbed: An Overview, ACM International Conference Proceeding Series; Vol. 50, Proceedings of the 5th international conference on Electronic commerce, 2003.

[85] HL7 Message Maker. http://www.itl.nist.gov/div897/ctg/messagemaker/, last visited on June 2009.