An Interoperability Test Framework for HL7-Based Systems

Tuncay Namli, Gunes Aluc, and Asuman Dogac

***Abstract*—Health Level Seven (HL7) is a prominent messaging standard in the eHealth domain, and with HL7 v2, it addresses only the messaging layer. However, HL7 implementations also deal with the other layers of interoperability, namely the business process layer and the communication layer. This need is addressed in HL7 v3 by providing a number of normative transport specification profiles. Furthermore, there are storyboards describing HL7 v3 message choreographies between specific roles in specific events. Having alternative transport protocols and descriptive message choreographies introduces great flexibility in implementing HL7 standards, yet, this brings in the need for test frameworks that can accommodate different protocols and permit the dynamic definition of test scenarios. In this paper, we describe a complete test execution framework for HL7-based systems that provides high-level constructs allowing dynamic set up of test scenarios involving all the layers in the interoperability stack. The computer-interpretable test description language developed offers a configurable system with pluggable adaptors. The Web-based GUIs make it possible to test systems over the Web anytime, anywhere, and with any party willing to do so.**

***Index Terms*—Collaborative eHealth processes, health level seven (HL7), interoperability and conformance testing.**

I. Introduction

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].

Interoperability is a major challenge of today's eHealth applications. Several standards have been developed and some are still under development to address the various layers in the interoperability stack. 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 eHealth applications can be guaranteed and products can be certified.

Manuscript received February 1, 2008; revised August 11, 2008. Current version published May 6, 2009. This work was supported in part by the Scientific and Technical Research Council of Turkey (TÜBİTAK), TEYDEP Project No: 7070191 and in part by the Ministry of Health, Turkey.

The authors are with the Department of Computer Engineering, Software R&D Center (SRDC), Middle East Technical University (METU), Ankara, Turkey (e-mail: tuncay@srdc.metu.edu.tr; gunes@srdc.metu.edu.tr; asuman@srdc.metu.edu.tr).

Digital Object Identifier 10.1109/TITB.2009.2016086

One of the most important family of standards in eHealth is Health Level Seven (HL7) [3]. In fact, HL7 v2.x [4] is the most widely used eHealth message standard in the world today.

However, any HL7 implementation must also address the other layers of interoperability, namely the business process layer (describing the message choreographies) and the communication layer (specifying the transport standards and protocols). This need has already been recognized by the HL7 standard developers and HL7 v3 provides a number of normative transport specification profiles, namely the ebXML message specification profile [5], the web services profile [6], and the transmission control protocol/Internet protocol (TCP/IP)-based minimal lower layer protocol profile [7]. For the business process layer, there are storyboards that describe HL7 v3 message choreographies between specific roles in specific events [8]. However, the storyboards are descriptive rather than normative.

Refraining from specifying strict standards for communication and business process layers of interoperability, as HL7 does, has the advantage that it allows a wide variety of implementation techniques, and hence, provides flexibility and ease in implementations. However, being able to use different protocols and standards to implement HL7-based systems necessitate a dynamic and configurable test framework.

In this paper, we introduce an interoperability test framework to test all the layers in the interoperability stack [9] for the HL7-based systems, namely the Testing Business Process, Application, Transport, and Network Layers (TestBATN) framework. The system has the following features.

1) A test execution model consisting of high-level test constructs that can handle or simulate different parts or layers of the interoperability stack. This makes it possible to test different scenarios based on different standards by generating the corresponding test scenarios and plugging in different adaptors.
2) A computer interpretable test description language allowing dynamic set up of test cases. This provides flexibility to design, modify, maintain, and extend the test functionality in contrast to *a priori* designed and hard-coded test cases.
3) Ability to automate the whole testing process addressing all the layers of the interoperability stack. Partly automating the test process, i.e., providing tools for certain layers of the interoperability stack and doing the rest manually results in human labor intensive, error prone, costly to develop test processes.
4) The system aims for "low cost of entry" for the test participants, and hence, provides a graphical environment where a test designer can assemble the reusable test constructs for conformance and interoperability tests.

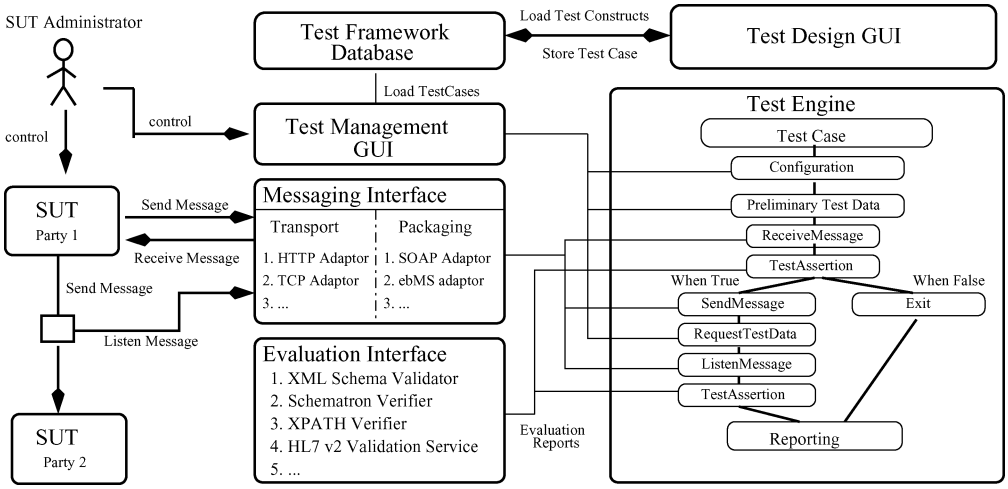

Fig. 1. Overall architecture of the TestBATN framework.

5) The TestBATN framework is Web-based. At the age of Internet, interoperability testing should not be restricted in time and place. Vendors should be able to test their products over the Web anytime, anywhere, and with any party willing to do so.

The paper is organized as follows: Section II gives an overview of the system architecture explaining the motivations behind the design and implementation decisions. The various layers of interoperability testing and the corresponding TestBATN framework components are also described in this section. Section III covers the test description language constructs. Section IV presents the user experience on the tests realized through TestBATN. TestBATN framework is currently being used by the Ministry of Health (MoH), Turkey, to test the conformance and interoperability of vendor applications to National Health Information System (NHIS), Turkey. NHIS is based on HL7 v3 messages and HL7 web services profile. In Section V, the TestBATN framework's further functionality is explained through a future scenario of NHIS, Turkey. Section VI describes the implementation tools and techniques. Section VII covers the related work and summarizes a number of recent research and development efforts on test tools, languages, and frameworks that has influenced the TestBATN framework design. Finally, Section VIII concludes the paper.

II. TestBATN Framework

TestBATN framework is designed to allow dynamic, configurable, and fully automated execution of conformance and interoperability test scenarios. Conformance to a standard is a system's ability to satisfy all the requirements expressed within the standard. Conformance testing is realized through a collection of test cases each evaluating whether the requirements are satisfied. On the other hand, interoperability testing involves not only the conformance evaluation for each role, but also testing their ability to function collaboratively to achieve a set of requirements [10].

The overall architecture of the TestBATN framework is given in Fig. 1. The main components of the system are as follows.

1) *Messaging interface:* A messaging interface is used to test communication layer interoperability. In order to address the challenge of handling different standards and protocols used at the communication layer, two common messaging interfaces are developed, namely the transport interface and the packaging interface through which various adaptors can be registered to the system and used.
2) *Evaluation interface:* Evaluation interfaces are used at the document interoperability layer for syntactic validation and semantic verification of HL7 messages and documents. There are several different validation and verification adaptors provided by the TestBATN framework such as XML schema validators or XPATH [11] verifiers.
3) *Test engine:* Test engine drives the test scenario defined through the TestBATN test description language by communicating with the system under test (SUT) administrators, evaluation interfaces, and messaging interfaces as needed. It acts like an interpreter and executes test steps defined in the test case XML instance according to the flow constructs. During this interpretation, the test engine maintains an environment that consists of variables, expressions, and placeholders.
4) *Test design GUI* and *test management GUI:* To facilitate the test design and monitoring Web-based graphical tools are provided to the users.
5) *Test framework database* stores all the reusable test material including the test descriptions and the adaptors.

In the following sections, we describe how this architecture handles testing the various layers of the interoperability stack.

A. Communication Layer Testing

The communication layer in the interoperability stack addresses the *transport protocol* such as hypertext transfer protocol (HTTP) [12] and the *packaging protocol* such as "simple object access protocol (SOAP)" [13].

As already mentioned, the HL7 v2.x standard does not mandate a protocol for managing networks. Rather, it recommends several protocol alternatives, called "lower layer protocols"

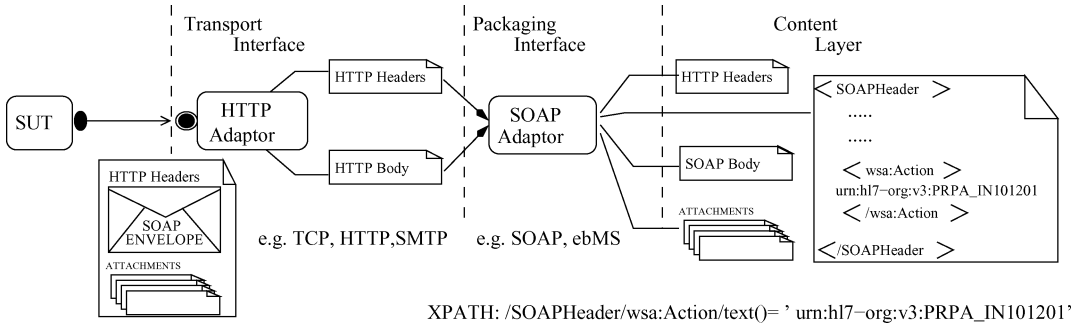

Fig. 2. Overview of the TestBATN messaging interface.

for the transmission of the messages [14]. The recommended transport protocols are "minimal lower layer protocol" based on TCP/IP [15], "hybrid lower layer protocol" based on RS-232 [16], and ANSI X3.28-based data link protocol [17]. There are also implementations using bare TCP/IP and FTP.

Contrary to the HL7 v2.x, HL7 v3 provides a number of transport specification profiles and specifies them as normative. These profiles are ebXML message specification profile [5], web services profile [6], and TCP/IP-based minimal lower layer protocol profile [7].

It is clear that in order to test HL7-based systems at the communication layer, a test framework must support these various different transport and communication protocols. Our solution to this challenge is to define two common interfaces: the transport interface and the packaging interface (Fig. 1). Then, various adaptors are developed and registered to the system conforming to these interfaces.

The adaptors in the transport interface have the ability to open or use existing server- or client-side connections such as an HTTP connection for an HTTP adaptor or a TCP connection for a TCP adaptor. Additionally, they can break down a message into fragments such as "HTTP header" and "HTTP body" so that further tests can be specified on the fragments, as shown in Fig. 2. It is also possible to construct a message from the given fragments.

Similarly, the packaging is handled through one of the adaptors registered according to the packaging interface, such as, a SOAP adaptor or an ebMS adaptor. Although these adaptors do not need to have the networking capabilities, they have the ability to fragment a message into its parts such as *SOAP header*, *SOAP body,* and *SOAP attachments* or defragment the parts into a message according to the standard they support. The fragments are assigned to variables so that further test assertions can be specified and tested.

Both the transport and the packaging adaptors are configurable, i.e., it is possible to apply them selective restrictions. For example, the HTTP adaptor can be configured with the HTTP method parameter "GET" in order to accept only "HTTP GET" messages.

As another example, HL7 Web service profile [6] sets some restrictions over SOAP header part of the Web service communication regarding the Web Services (WS)-Addressing [18] specification stating that *wsa:MessageID*, *wsa:ReplyTo*, *wsa:To*, and *wsa:Action* elements must exist in SOAP header. The TestBATN framework allows designing a test case where the SOAP adaptor fragments the captured SOAP message and produces a fragment including only the SOAP header part. Then, XPATH expressions over this XML fragment are specified to test both the existence of the WS-Addressing elements and the header values to validate whether they are used correctly according to the system configurations.

```
<sch:rule context='cda:ClinicalDocument'>
...
<sch:assert test="cda:recordTarget[cda:patientRole
  [cda:patient[cda:name[cda:given = 'John']]]]"> ERR:
  recordTarget/patientRole/patient/name/given should be John
</sch:assert>
<sch:assert test="cda:recordTarget[cda:patientRole
    [cda:patient[cda:name[cda:family = 'Doe']]]]"> ERR:
    recordTarget/patientRole/patient/name/family should be Doe
</sch:assert>
...
</sch:rule>
```

Fig. 3. Example schematron to test a given HL7 CDA document for patient demographics information.

B. Document Layer Testing

Document layer interoperability testing is achieved through the evaluation interface component, as shown in Fig. 1. The semantics of each function of each of the adaptors conforming to this interface, the number of input parameters, the types of input parameters in terms of the data types are specified during the registration of the evaluation adaptors.

For example, assume that an HL7 clinical document architecture (CDA) [19] conformance test suite is to be developed for a clinical document management system and one of the test cases is checking whether the SUT handles the patient demographic information in the HL7 CDA documents correctly. The SUT is given the patient demographics information and is requested to generate the corresponding CDA document. After receiving the document from the SUT, the test engine uses a Schematron [20] validator to test the patient demographics part of the CDA document. For example, the Schematron shown in Fig. 3 tests whether the name and surname of the patient is correct in the document. When the assertions fail, error reporting entries are generated to be included into the test case report.

C. Business Process Layer Testing

A scenario-based test aims to verify the application behavior and responses according to an eHealth collaboration scenario. HL7 does not define normative collaboration processes. The storyboards for HL7 v3 messages describe illustrative message choreographies. Therefore, a test framework for HL7 must give its users the ability to define any real life scenario to be tested. In the TestBATN framework, test scenarios can be dynamically specified using the test design GUI and the associated test description language.

The TestBATN framework provides the following functionalities at this layer:

1) Not all the applications representing the roles to be tested in a scenario may be present in the test environment. These application behaviors are simulated through the messaging and data management capabilities of the TestBATN framework.
2) In the scenario-based tests, there is a need to capture and test the message exchanges among the SUTs playing different roles. For this purpose, a proxy mechanism is designed as a mediator which listens to the messages between the systems. The test engine plays the mediator role by using the receiving and sending capabilities of the corresponding Messaging Adapters in a special proxy mode.
3) To test the message choreography, it is necessary to check that the sender/receiver parties actually correspond to the roles claimed in the collaboration process and that the messages are exchanged in the correct order. To achieve this functionality, the TestBATN framework utilizes a twofold solution:
    a) Systems registered to the framework are provided with a certificate, which they use in all of their transactions with the test engine, to achieve Transport Layer Security (TLS) [21]. This way, the test engine can authenticate and correctly identify the initiator of the transaction.
    b) The test engine assigns different ports to transactions. Therefore, it is possible to explicitly differentiate between the transactions based on port numbers.
4) Trigger Events are another notion in business process definitions that the TestBATN framework supports. HL7 defines trigger events for each of its transactions. These are categorized as interaction-based, state-transition-based and user request-based trigger events. Interaction and state-transition-based trigger events are implicitly represented through the business processes that the users define. On the other hand, user request-based trigger events are handled by giving instructions to the SUT administrators through the TestBATN framework's GUIs.

D. Graphical User Interfaces

The test management GUI and the test design GUI are the two interfaces that complement the test description language and the test execution environment.

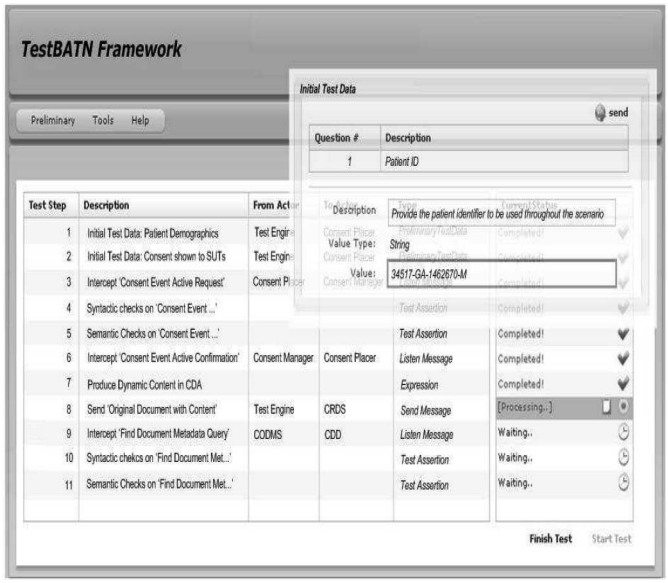

Fig. 4. Snapshot of the test management GUI.

The test management GUI is an interactive, multi user, event-driven, Web-based monitoring, and management environment for test executions in the TestBATN framework. It provides a medium for the SUT administrators to monitor the test execution while controlling their own systems based on the instructions received from the test engine. These instructions are performed in an event-driven way during configuration management, as well as during the initialization and handling of the test data.

Before the test execution, the scenario represented as a list of test step descriptions is conveyed to the SUT administrators, as it is illustrated in the snapshot given in Fig. 4. This information is collected from individual test step definitions provided during the test design phase. This gives the SUT administrator the ability to examine the whole testing process and the sequencing and branching among the test steps. After the test is initiated, the administrator can monitor the execution, i.e., he can track the steps that are completed or that are currently being processed. In the end, a report displaying the results of the test assertions and an error log for each test is generated and depicted to the administrator.

The test design GUI is a user-friendly interface to facilitate a test harness design, i.e., the set up for a test case representation through reusable language components and plug-ins. The test design GUI enables users to design a test scenario from two different views: the business process view and the test harness view. The former concentrates on the business process upon which the test scenario is based and the latter on the steps associated with the test semantics.

In the business process view, as shown in the Fig. 5, the designer describes a business process graphically by identifying the participating systems, the roles the systems play in the scenario and the transactions among the roles.

Note that as a future work, we plan to use the business process modeling notation (BPMN) [22] for describing the choreography of the test scenarios.

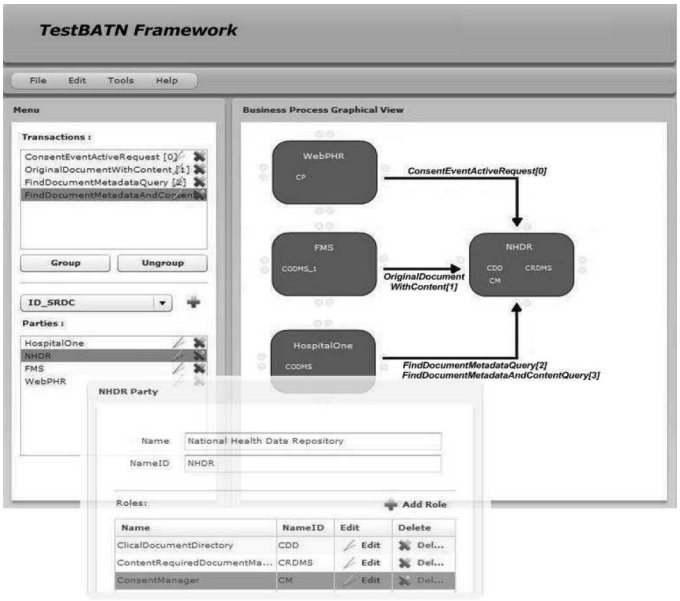

Fig. 5. Snapshot of the test design GUI—business process view.

After the designer specifies the systems to be tested as well as the simulated systems, the corresponding constructs in the test description language are automatically generated and the design process continues with the test harness view.

In the test harness view, the test designer defines test assertions and specifies its related messaging details. The test design GUI, as shown in Fig. 6, uses the semantic metadata associated with the registered adaptors in assisting the designer to conveniently assemble the test harness from these pluggable adaptors. Furthermore, the test designer GUI provides the supplementary tools (e.g., Schematron Editor, XML Editor) to facilitate the manipulation of the content utilized by the adaptors.

III. Test Description Language

The main component of the language is a *TestCase* construct, as given in [23], which is made up of the following constructs.

1) *Constructs to handle configuration information:* The TestBATN framework supports not only the compile-time, but also the run-time system configuration through three major mechanisms.
   a) *Handling network configuration information:* The *CommunicationConfiguration* construct is used to prompt the participating systems so that they can provide their configuration values for the communication- and transport-related parameters such as the endpoints and the ports used in the transaction.
   b) *Preliminary test data handling:* The *PreliminaryTestData* construct is used for the initialization of the necessary variables at the beginning of the *TestCase* execution. When a user specifies the initial test values, the other parties in the scenario are immediately notified. As an example, a participant may set the patient identifier used in the exchanged messages prior to *TestCase* execution, and all the involved parties are informed of this.
   c) *Request test data*: During the test execution, the scenario may involve cases where the value of an initialized variable needs to be modified by the administrators of the participating systems. The *RequestTestData* construct is used for this purpose.
2) *Variable declarations, expressions, and placeholders:* In the TestBATN framework, "Variables" are used to increase internal and external reusability of data and to have a highly adaptable run-time behavior. Whether the TestBATN framework simulates an application behavior, or information is received from an external source, it is stored in variables. All types of information like messages, documents and their fragments can be stored in variables. The TestBATN framework also employs variables for preliminary test data processing, for handling configuration information, and as placeholders in content customization.

   Expressions, on the other hand, are used to facilitate the manipulation of variables. It is possible to use XPath and JavaScript functions in the expressions. Furthermore, since expressions can be used to dynamically modify variable values, run-time type checking capability is also supported by the language.

   Placeholders make information gathered during the test execution available to message or document generation. For example, for a given transaction, there may be a document template available to the test designer fulfilling the syntactic requirements. The test designer can put placeholders in this template to customize its semantics. The placeholders simply act as variable references placed inside the template so that the run-time values of the variables are replaced with that of the placeholder.

   A further use of the variables, expressions, and placeholders is in realizing the simulated application roles. In simulating the functionalities of such roles, the TestBATN framework utilizes these constructs to store, partition, and update data received from other applications on behalf of simulated roles.
3) *Threads:* In order to support the concurrent execution of the groups of test steps, the "thread" concept of the programming languages is used. Since threads are atomic, reusable entities in terms of functionality, the test designer may rearrange them to express different test scenarios. The threads that are bound to run concurrently are arranged into a ThreadGroup.
4) *Test steps:* Threads are composed of logical subunits called the *TestSteps* that are organized to perform a specific task. This particular arrangement enables explicit reporting for each step. A *TestStep* may realize two roles: that of a communicator through the use of the constructs *ReceiveMessage*, *SendMessage*, or *ListenMessage*, or that of an evaluator using the construct *TestAssertion*.
   a) *ReceiveMessage* is a test step that instructs the test engine to initiate a specific set of messaging adaptors (the transport and the packaging adaptors) to receive a message from a SUT. A

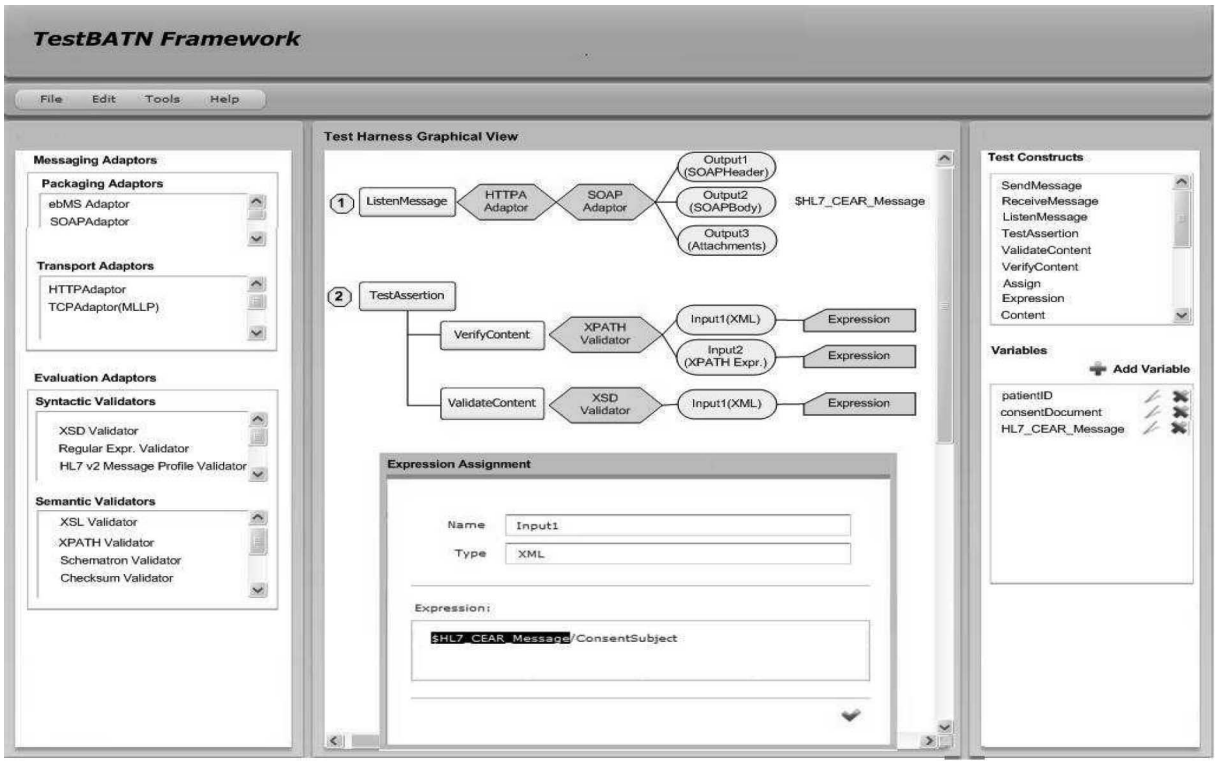

Fig. 6. Snapshot of the test design GUI—test harness view.

ReceiveMessage step simulates only one way of the transaction, i.e., the receiving side while receiving the request message or while receiving the response message.

b) *SendMessage* is the test step which complements *ReceiveMessage* step for the two-way transactions and instructs the test engine to initiate a specific set of messaging adaptors (the transport and the packaging adaptors) to send a message to a SUT. By using *ReceiveMessage* and *SendMessage* constructs in a test case, the test engine can simulate a role that receives a request from a SUT and responds to it; or a role that sends a request and receives the response from the SUT. Note that any number of other test steps can be executed between these steps since they are correlated with each other by using a transaction identifier.

c) During the scenario tests, there is also a need to listen to the communications between the SUTs. *ListenMessage* is the test step that instructs the test engine to initiate the specific messaging adaptors to listen and forward a message from a SUT to another SUT. During the execution of this step, message adaptors run as proxies between the two SUTs. Therefore, while the content of the message is stored so that later test steps in the test case can use it, the message is also transmitted to the intended receiver (another SUT). Similar to *ReceiveMessage* and *SendMessage*, *ListenMessage* also corresponds to one way communication. Another complementary *ListenMessage* is needed to listen to the response of the SUT.

d) *Test assertion:* Test assertion steps are used for semantic and syntactic validations of the information received from SUTs by messages or through user interactions. Each test assertion step corresponds to a set of constraints or requirements of the standard or interoperability profile that the SUTs are expected to conform.

The pluggable nature of the adaptors provides a high degree of configuration capability to the test steps. Furthermore, the adaptors themselves may be configured by giving the appropriate attribute values. For example, by setting the HTTP method parameter of the HTTP adaptor either to "GET" or to "POST," it is possible to configure the communication.

5) *Flow constructs for branching, looping, and concurrent execution:* These constructs give test writers the ability to control the flow of the scenario and also to deal with failure conditions, and hence, to have dynamic control over the execution sequence.

6) *Handling HL7 v.2 messages:* It is difficult to measure compliance of HL7 v2.x interfaces because of the many optional features. For conformance and interoperability testing, the interfaces must have precise and unambiguous specifications. To handle this problem, a "message profiling" methodology [24] is introduced for HL7 [25]. Message profiles constrict the definition of a message such that the optional constructs and processing rules of a message are explicitly stated. In the TestBATN framework,

HL7 v2 message testing is based on the message profiles specified by the user.

Furthermore, HL7 v2 messages are electronic data interchange (EDI) [26] based. We use HL7 application programming interface (HAPI) assembler/disassembler tool [27] to transform the HL7 v2 EDI messages into their XML representations so that XML tools such as XPATH expressions or Schematrons can be used with them in the TestBATN framework.

IV. Real Life Usage of TestBATN: Testing the Conformance and Interoperability of Vendor Applications to Turkey's NHIS

The flexibility in implementing HL7 may also become an obstacle in achieving interoperability especially for large-scale integration efforts such as a nation-wide implementation. Providing interoperability profiles as recommended by HL7 and extensive testing of system components for conformance and interoperability are necessary.

For example, in Turkey, HL7 v3 messaging standards are used nation wide to convey patient care information from Family Medicine Information Systems (FMISs) [28] to the National Health Data Repository (NHDR) [29]. TestBATN is currently being used to test the conformance and interoperability of the FMISs and Hospital Information Systems (HISs) to Turkey's NHIS.

The NHIS, Turkey provides 23 Web services each specialized to transmit a specific Minimum Health Data Set (MHDS) [30] (called a "transmission schema") based on the following national and international standards: for transport protocol, HL7 web services profile [31] is used together with WS-Security [32] username token over SSL. The "transmission schemas" used in Web services are HL7 CDA R2 [19] compliant EHRs and each HL7 CDA section is a MHDS [33] that is formed from the data elements specified in the National Health Data Dictionary (NHDD) [34]. Health Coding Reference Server (HCRS) [35] serves all coding systems in use in Turkey that are used in the data elements within the MHDSs. For some specific data elements, some other coding systems, like ICD-10 coding system [36], are used. For each transmission schemas and MHDS, 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) [37]. 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 [38]. The id numbers of the doctors used in the messages should exist in this system.

As described in [39], TestBATN scenarios are developed implementing the testing requirements of this system, consisting of syntactic validation steps, validation against code lists and business rules, and semantic validation based on preliminary test data. The TestBATN test description language together with the test design GUI reduced the human labor required to develop 200 test scenarios categorized under 25 test suites to less than two person-months indicating that the test scenarios are easy to configure.

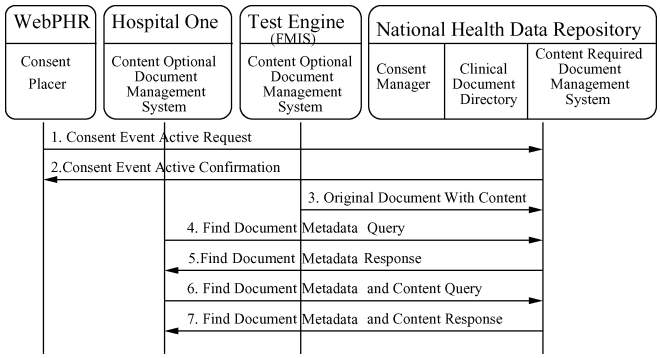

Fig. 7. Example of national profile.

The TestBATN services for NHIS test scenarios are publicly available from MoH, Turkey servers [40] with more than 60 active users daily. Also, MoH, Turkey organized a "Connect-a-thon" like event, a five days workshop in June 2008. The aim was to bring together all the vendors in a physical location to test their products to speed up the integration with NHIS. During the workshop more than 5000 test scenarios are executed through TestBATN by an average of 130 participants from 55 vendors. This implies that the system reduced the effort needed for testing considerably. At the end of the workshop, each vendor is provided with a very detailed report on the test scenarios and steps performed. These reports are automatically produced by TestBATN tools. The survey conducted at the end of the workshop revealed a very high level user satisfaction with the testing system indicating that testing system is acceptable by the administrators of the SUT.

V. Example Scenario Execution Demonstrating TestBATN Functionalities

Currently, NHIS is mostly used for decision support within the MoH [41]. However, in the future the healthcare organizations as well as the patients themselves will be able to query and retrieve EHRs from this repository when the system is coupled with a patient consent management component to provide privacy and security to the healthcare data.

To implement this future scenario, the HL7 v3 interoperability profiles based on the "NHDR" [42] and "data consent" [43] storyboards are necessary with fine-grained constraints imposed by the nation's healthcare policies.

As shown in Fig. 7, the HL7 v3 roles in this profile are the "consent placer" [44] played by a Web Personal Health Portal ("WebPHR") and the "content optional document management system" [45] played by a healthcare organization ("HospitalOne"). Furthermore, since NHIS NHDR functions both as a repository for health documents and as a reference point for consent activation and enforcement, three roles have been associated with it: "clinical document directory" [46], "content required document management system" [47], and "consent manager" [48]. Test engine, on the other hand, plays the role of

any healthcare organization, such as a FMIS, that can submit or receive patient care information.

In this profile, the consent of a patient on his EHR is stored in the NHDR via the interaction between the "consent placer" and the "consent manager" through transactions 1 and 2 (Fig. 7). Furthermore, the profile defines HL7 CDA templates for several document types such as "discharge summary." Submission, registration, query and retrieval of EHRs and their metadata are achieved through transactions 3–7 (Fig. 7). There are syntactic and semantic restrictions on the formats and payloads of these transactions. Finally, the profile specifies fine-grained access control rules on the EHR documents.

Such a test scenario is executed in the TestBATN framework as follows.

1) First, there is a need to inform all the parties about the initial test data. In the given scenario, the initial test data are the patient identifier and the patient demographics information. In the TestBATN framework, these data are made available through the *PreliminaryTestData* construct. However, the test designer may wish to design a more generic test scenario instead of restricting the tests to some specific patient information. For example, assume that the test designer appoints the administrator of the PHR system to be in charge of determining the patient information. Then, the test engine requests the PHR administrator to specify the necessary information by using the *PreliminaryTestData* construct. Once this information is made available, the related parties register the patient to their systems.
2) The example profile requires that the applications playing the HL7 consent placer role should be able to produce both fine-grained and record-based consents. In order to test if the PHR system has this capability, the administrator of the PHR system is requested to produce a consent document based on the access control descriptions provided by the test writer. An example access control description can be: "Register consent such that mental illnesses of the patient listed in the "ProblemList" section of the Discharge Summary document are masked and shown solely to the patient's Psychologist."
3) The administrator of the PHR system gets the rule sent by the test engine and produces the consent based on the given requirements. Consequently, the "consent event activate request" [49] message is constructed with the consent as its payload and sent to the NHDR. The test engine intercepts this transaction to check if the consent produced complies first with the profile (syntactically—e.g., the XSD adaptor) and then with the rule definition given earlier (semantically—e.g., Schematron validation, XPath evaluation). If the NHDR registers the consent successfully and the test engine recognizes the response as the "consent event activate confirmation" [50] message, the test continues. Otherwise, the test is aborted with a report on the failure. The branching and looping constructs in the TestBATN framework enables users to define such test scenarios.
4) In the next step, the test engine behaves as the "content optional document management system" and initiates "original document with content" [51] transaction. The template CDA document provided by the test designer is tailored at run-time according to the semantics of the scenario. Placeholders present in the template document make this dynamic functionality possible. The test engine substitutes these placeholders with their run-time values such as the patient identifier information initialized by the PHR administrator at runtime. The test engine registers various EHR documents, one of which is of type "discharge summary," to the NHDR.
5) In this step, the administrator of "HospitalOne" queries NHDR for all "discharge summary" documents of the given patient. The "Find Document Metadata Query" [52] and its response messages are intercepted by the test engine for further syntactic and semantic processing. More specifically, the query parameters are tested to check whether they are set correctly for the document type. In this way, not only the exchanged documents are validated for conformance, but also it is possible to determine if the parties provide the expected functionalities.
6) Following the previous interaction, the administrator of "HospitalOne" authenticates himself to the system as the patient's "General Practitioner" and issues a "Find Document Metadata and Content Query" [53] to retrieve the desired document. First, the Test Engine verifies that the document is actually the one that has been registered. The "Checksum Validator" is an adaptor registered to the TestBATN framework that enables such tests. How NHDR handles the patient consent must also be tested. According to the given access restrictions, the "ProblemList" section in the document should be masked if the access requestor is not a "Psychologist." Since the role is a "general practitioner," the "ProblemList" section in the returned document can be checked through an XPath expression to see if it is masked for the mental illnesses of the patient.
7) According to the profile, the "content optional document management system" application role should properly render the CDA document received. Since the test designer is the producer of the document, by requesting information from the SUT about the document content through the *RequestTestData* construct and comparing the answers with the data contained in the variables through *TestAssertions*, the ability of the SUT to correctly render the document can easily be tested.

VI. Implementation Tools and Techniques

In the TestBATN framework, the test engine and the adaptors of the messaging and evaluation interfaces are developed in Java Platform SE (version 1.6.0) [54]. The GUIs are implemented using Adobe Flex 2.0.1 [55]. Adobe Flex provides a collection of rich Internet application development technologies for rendering user-friendly and dynamic applications.

The multithreading functionality provided by the Java APIs enables multiple users to connect and to monitor the execution

of different test cases simultaneously. In this multiuser environment, thanks to the Adobe Flex server data-push capability, changes made in one test management GUI (e.g., a user setting the value of a variable prompted through the *PreliminaryTestData* interaction) are seamlessly distributed to the others, as the test engine pushes these events onto the communication channel.

TestBATN GUIs are based on the model-view-controller architectural pattern. Furthermore, Adobe Flex enables the view to be bound to the model through event dispatching mechanisms. Consequently, user notifications can take place instantaneously.

The test engine is constructed as a layered architecture. More specifically, the messaging interface and the evaluation interface have been built as separate components. This gives the developers of the TestBATN framework the ability to produce new communication and validation adapters and register them to the system easily.

VII. Related Work

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

One of the first and most successful work on test automation is the *Testing and Test Control Notation* (TTCN) [56] standard published by the European Telecommunications Standard Institute (ETSI) [57]. The latest version, TTCN-3 is a computationally complete programming language for expressing test cases for conformance and interoperability testing in the telecommunication domain. Due to the nature of testing problems in the telecommunications domain, TTCN focuses on the details of communication but does not fully represent business processes.

The OASIS IIC ebXML test framework [58] is developed to support conformance and interoperability testing for ebXML specifications. It describes a testbed architecture and its software components as well as how these can be combined to create a test harness for various types of testing. Since this framework is based on ebXML messaging, it focuses on its message service handler (MSH) implementations instead of application testing. However, the test description language contains high-level constructs that sufficiently describe the eBusiness test processes.

A sample implementation of ebXML TestFramework 1.1 is provided within the National Institute of Standards and Technology (NIST) B2B Testbed [59] project. NIST B2B Testbed is initiated by the NIST [60] to develop software tools that can be used to test current B2B standards or technologies and enhance the capability for on-demand demonstration of conformance and interoperability [10]. They conducted several interoperability demonstrations together with the STAR/XML initiative [61] from the automotive industry and with the US Air Force.

OASIS event-driven test scripting language (eTSL) [9], 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 message choreographies, and the business document standards.

The TestBATN framework is influenced both by OASIS IIC ebXML test framework and by eTSL, and complements them by providing a complete test framework specialized to eHealth together with its adaptors. In the TestBATN framework, the event notion in eTSL is extended to capture the requirements of the test scenarios encountered in the eHealth domain. These requirements include interaction with the users of an 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. 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.

There are also some eHealth-specific testing tools. Currently, HL7 together with NIST [60] started an initiative to automatically and dynamically generate test messages, called HL7 Message Maker [62]. The data used to populate the messages are drawn from a number of sources including the NIST-developed database of HL7 data items, HL7 tables, user tables, and external tables.

As a future work, this study in progress will be used in the TestBATN framework to dynamically create meaningful test data without compromising patient confidentiality and privacy. In other words, the preliminary test data handling functionality will be enhanced so that the values of variables will be selected dynamically from a data source. For example, assume the test designer wishes to restrict a diagnosis element, expressed with ICD-10 codes, to a specific value in the test scenario. Currently, in the TestBATN, this requirement is shown to the user during preliminary test data handling phase and the subsequent test assertions validate this restriction. However, being able to randomly choose a value from a set of relevant values defined by the domain expert will improve the system. As a result, the test case definitions will behave as scenario templates and in each execution the SUT will be presented with a different restriction.

IHE integration profiles [63] are tested through MESA tools [64]. MESA tools consist of two parts: tools simulating an actor's functionalities that are written in a programming language (C++ or java) and a test definition with a scripting language (Perl) for each test specified. Companies before participating to a Connect-a-thon download MESA tools and run the test scripts that are written for the actor that their system claims to be conformant with. Test scripts use the tools to simulate the corresponding actor and send or receive messages from the SUT. During the execution test scripts produce relevant messages (error or log) and collect them into log files.

Managing a Connect-a-thon is a human labor intensive task: most of the testing process is conducted by humans such as checking message contents or application behaviors. There is also an on going project by IHE to improve Connect-a-thons. Nevertheless, the TestBATN framework can be used to capture the testing requirements of IHE integration profiles since the TestBATN framework addresses all the layers of the interoperability stack as IHE does. In fact, the scenario described in Section V can also be implemented using the "cross enterprise document sharing (XDS)" [65] and the "basic patient privacy

consent (BPPC)" [66] profiles of IHE, and we have demonstrated that TestBATN has the capabilities to test such scenarios.

VIII. Conclusion

Interoperability standards often contain certain ambiguity in their specifications that may result in differences in their implementations. Therefore, conformance and interoperability testing are very important to maintain correct information exchange. This problem becomes more challenging in the eHealth domain because there are many standards for each layer of interoperability and also the healthcare scenarios are much more complex than B2B scenarios.

The complete test framework introduced in this paper allows stakeholders to perform conformance and interoperability tests for their products based on HL7 standard specifications. The framework is built up on an extensible test execution model where the model is represented through a computer interpretable test description language. The execution model and the language support the flexibility to design, modify, maintain, and extend the test functionality as the evolving requirements necessitate, in contrast to the a priori hard coded test cases. The whole testing process is automated, hence significantly reducing human labor and the associated costs. Also, the graphical Web-based environment provided helps to establish a "low cost of entry" for the test designers and participants.

The hands-on experience obtained in implementing the test scenarios for Turkey's NHIS demonstrated the effectiveness of localized HL7 profiles constraining the business processes, communication and messaging details. In this respect, we believe that it will help if HL7 v3 produces formal business process profile templates for its affiliates to facilitate achieving interoperability at this layer.

References

[1] IEEE Dictionary, *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*. New York: IEEE, 1990.

[2] N. Brown and M. Reynolds, "Short Strategic Study CEN/TC 251/N00-047: Strategy for production and maintenance of standards for interoperability within and between service departments and other healthcare domains," CEN/TC 251 Health Informatics, Brussels, Belgium, 2000.

[3] Health Level 7 (HL7). (2009, Apr. 12). [Online]. Available: http://www.hl7.org/

[4] Health Level 7 v2.x Standards. (2009, Apr. 12). [Online]. Available: http://www.hl7.org/Library/standards.cfm

[5] ebXML Messaging Services Profile. (2008, Jul.). [Online]. Available: http://www.hl7.org/v3ballot/html/infrastructure/transport/transport-ebxml.htm (accessed April 12, 2009).

[6] Web Services Profile. (2009, Jan.). [Online]. Available: http://www.hl7.org/v3ballot2009jan/html/welcome/environment/index.htm (accessed April 12, 2009).

[7] Minimal Lower Layer Protocol Profile. (2009, Mar.). [Online]. Available: http://www.hl7.org/v3ballot/html/infrastructure/transport/transport-mllp.htm (accessed April 12, 2009).

[8] HL7 V3 Guide. (2009, Mar.). [Online]. Available: http://www.hl7.org/v3ballot/html/help/v3guide/v3guide.htm (accessed April 12, 2009).

[9] (2007, Nov.). Event-Driven Test Scripting Language (eTSL). OASIS ebXML Implementation Interoperability and Conformance (IIC) TC, Working Draft 0.85. [Online]. Available: http://www.oasis-open.org/apps/org/workgroup/ebxml-iic/download.php/26036/eTSL-draft-085.pdf

[10] B. Kulvatunyou, N. Ivezic, M. Martin, and A. T. Jones, "A business-to-business interoperability testbed: An overview," presented at the 5th Int. Conf. Electron. Commerce (ACM International Conference Proceeding Series, vol. 50), Pittsburgh, PA, 2003.

[11] XML Path Language. (1999, Nov.). [Online]. Available: http://www.w3.org/TR/xpath (accessed April 12, 2009).

[12] Hypertext Transfer Protocol. (2009, Apr. 12). [Online]. Available: http://www.w3.org/Protocols/

[13] Simple Object Access Protocol (SOAP). (2007, Apr.). [Online]. Available: http://www.w3.org/TR/SOAP/ (accessed April 12, 2009).

[14] HL7 Lower Layer Protocols. [Online]. Available: http://www.hl7.org/Memonly/downloads/Standards_Messaging_v251/HL7_Messaging_v251_PDF.zip

[15] Transmission Control Protocol/Internet Protocol. (2009, Apr.). [Online]. Available: http://en.wikipedia.org/wiki/Transmission_Control_Protocol (accessed April 12, 2009).

[16] Recommended Standard 232. (2009, Apr.). [Online]. Available: http://en.wikipedia.org/wiki/RS-232 (accessed April 12, 2009).

[17] ANSI X3.28. (1997, Aug.). [Online]. Available: http://library.abb.com/GLOBAL/SCOT/SCOT229.nsf/VerityDisplay/D72E25504B0B2178C2256A720035BD36/$File/REC501ANSI_EN_A.pdf (accessed April 12, 2009).

[18] Web Services Addressing. (2004, Aug.). [Online]. Available: http://www.w3.org/Submission/ws-addressing/ (accessed April 12, 2009).

[19] HL7 Clinical Document Architecture. (2009, Mar.). [Online]. Available: http://www.hl7.org/v3ballot/html/infrastructure/cda/cda.htm (accessed April 12, 2009).

[20] Document Schema Definition Language (DSDL), Part 3: Rule-Based Validation—Schematron, ISO/IEC 19757-3:2006. (2009, Apr. 12). [Online]. Available: http://standards.iso.org/ittf/PubliclyAvailableStandards/c040833_ISO_IEC_19757-3_2006(E).zip

[21] Transport Layer Security. (2009, Apr.). [Online]. Available: http://en.wikipedia.org/wiki/Secure_Sockets_Layer (accessed April 12, 2009).

[22] Business Process Modeling Notation (BPMN). (2008, Jan.). [Online]. Available: http://www.bpmn.org/Documents/BPMN%201-1%20Specification.pdf (accessed April 12, 2009).

[23] An Example Test Case Construct of the TestBATN Test Description Language. (2009, Apr. 12). [Online]. Available: http://www.srdc.metu.edu.tr/testing/SampleTestCase.xml

[24] R. Snelick, P. Rontey, L. Gebase, and L. Carnahan, Towards Interoperable Healthcare Information Systems: The HL7 Conformance Profile Approach. (2009, Apr. 12). [Online]. Available: http://www.itl.nist.gov/div897/ctg/messagemaker/papers/IESA2007.rsnelick.pdf

[25] HL7 Version 2.5. (2009, Apr. 12). [Online]. Available: http://www.hl7.org/Library/standards_mem1.cfm#HL7%20Version%202.5

[26] Electronic Data Interchange. (2009, Mar.). [Online]. Available: http://www.unece.org/trade/untdid/welcome.htm (accessed April 12, 2009).

[27] HL7 Application Programming Interface. (2009, Apr.). [Online]. Available: http://hl7api.sourceforge.net/ (accessed April 12, 2009).

[28] Aile Hekimligi Bilgi Sistemi and T. C. Saglik Bakanligi. (2009, Apr. 12). [Online]. Available: http://www.saglik.gov.tr/AHBS/Default.aspx

[29] Ulusal Saglik Bilgi Sistemi and T. C. Saglik Bakanligi. (2009, Apr. 12). [Online]. Available: http://www.saglik.gov.tr/USBS/BelgeGoster.aspx

[30] Y. Kabak, A. Dogac, I. Kose, N. Akpinar, M. Gurel, Y. Arslan, H. Ozer, N. Yurt, A. Ozcam, S. Kirici, M. Yuksel, and E. Sabur, "The use of HL7 CDA in the national health information system (NHIS) of Turkey," presented at the 9th Int. HL7 Interoperability Conf. (IHIC'08), Crete, Greece, Oct.

[31] HL7 Web Services Profile. (2008, Jan.). [Online]. Available: http://www.hl7.org/v3ballot2008jan/html/infrastructure/transport/transport-wsprofiles.htm (accessed April 12, 2009).

[32] WS-Security Core Specifications. (2006, Feb.). [Online]. Available: http://www.oasisopen.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf (accessed April 12, 2009).

[33] Minimum Saglik Veri Setleri and T. C. Saglik Bakanligi. (2009, Apr. 12). [Online]. Available: http://www.saglik.gov.tr/USBS/BelgeGoster.aspx

[34] Ulusal Saglik Veri Sozlugu and T. C. Saglik Bakanligi. (2009, Apr. 12). [Online]. Available: http://www.saglik.gov.tr/USBS/BelgeGoster.aspx

[35] The Health Coding Reference Server (HCRS). (2009, Apr. 12). [Online]. Available: http://sbu.saglik.gov.tr/SKRS2%5FListesi/

[36] International Classification of Diseases (ICD 10). (2009, Apr. 12). [Online]. Available: http://www.who.int/classifications/icd/en/

[37] MERNIS, Central Demographics Management System. (2009, Apr. 12). [Online]. Available: http://www.nvi.gov.tr/Hakkimizda/Projeler,Spot_Mernis.html

[38] Doktor Bilgi Bankasi and T. C. Saglik Bakanligi. (2009, Apr. 12). [Online]. Available: http://www.saglik.gov.tr/USBS/BelgeGoster.aspx?F6E10F8892433CFF7A2395174CFB32E1FAEB5097C9CF4D87

[39] T. Namli, G. Aluc, A. Sinaci, I. Kose, N. Akpinar, M. Gurel, Y. Arslan, H. Ozer, N. Yurt, S. Kirici, E. Sabur, A. Ozcam, and A. Dogac, "Testing

the conformance and interoperability of NHIS to Turkeys HL7 profile," presented at the 9th Int. HL7 Interoperability Conf. (IHIC'08), Crete, Greece, Oct.
[40] Ministry of Health, TestBATN Deployment. (2009, Apr. 12). [Online]. Available: https://195.142.107.246:8443/testexecution/TestExecutionGUI.html
[41] Karar Destek Sistemi and T. C. Saglik Bakanligi. (2009, Apr. 12). [Online]. Available: http://www.saglik.gov.tr/USBS/BelgeGoster.aspx
[42] HL7 v3 Ballot Site, Universal Domains, Medical Records Domain, Document Query Topic, National Health Data Repository Storyboard. (2009, Mar.). [Online]. Available: http://www.hl7.org/v3ballot/html/welcome/environment/index.htm (accessed April 12, 2009).
[43] HL7 v3 Ballot Site, Universal Domains, Medical Records Domain, Document Consent Topic, Data Consent Storyboard. (2009, Mar.). [Online]. Available: http://www.hl7.org/v3ballot/html/welcome/environment/index.htm (accessed April 12, 2009).
[44] HL7 v3 Ballot Site, Universal Domains, Medical Records Domain, Data Consent Topic, Application Roles, Consent Placer. (2009, Mar.). [Online]. Available: http://www.hl7.org/v3ballot/html/welcome/environment/index.htm (accessed April 12, 2009).
[45] HL7 v3 Ballot Site, Universal Domains, Medical Records Domain, Document Query Topic, Application Roles, Content Optional Document Management System. (2009, Mar.). [Online]. Available: http://www.hl7.org/v3ballot/html/welcome/environment/index.htm (accessed April 12, 2009).
[46] HL7 v3 Ballot Site, Universal Domains, Medical Records Domain, Document Query Topic, Application Roles, Clinical Document Directory. (2009, Mar.). [Online]. Available: http://www.hl7.org/v3ballot/html/welcome/environment/index.htm (accessed April 12, 2009).
[47] HL7 v3 Ballot Site, Universal Domains, Medical Records Domain, Document Query Topic, Application Roles, Content Required Document Management System. (2009, Mar.). [Online]. Available: http://www.hl7.org/v3ballot/html/welcome/environment/index.htm (accessed April 12, 2009).
[48] HL7 v3 Ballot Site, Universal Domains, Medical Records Domain, Data Consent Topic, Application Roles, Consent Manager. (2009, Mar.). [Online]. Available: http://www.hl7.org/v3ballot/html/welcome/environment/index.htm (accessed April 12, 2009).
[49] HL7 v3 Ballot Site, Universal Domains, Medical Records Domain, Data Consent Topic, Interactions, Consent Event Activate Request. (2009, Mar.). [Online]. Available: http://www.hl7.org/v3ballot/html/welcome/environment/index.htm (accessed April 12, 2009).
[50] HL7 v3 Ballot Site, Universal Domains, Medical Records Domain, Data Consent Topic, Interactions, Consent Event Activate Confirmation. (2009, Mar.). [Online]. Available: http://www.hl7.org/v3ballot/html/welcome/environment/index.htm (accessed April 12, 2009).
[51] HL7 v3 Ballot Site, Universal Domains, Medical Records Domain, Document Query Topic, Interactions, Original Document With Content. (2009, Mar.). [Online]. Available: http://www.hl7.org/v3ballot/html/welcome/environment/index.htm (accessed April 12, 2009).
[52] HL7 v3 Ballot Site, Universal Domains, Medical Records Domain, Document Query Topic, Interactions, Find Document Metadata Query. (2009, Mar.). [Online]. Available: http://www.hl7.org/v3ballot/html/welcome/environment/index.htm (accessed April 12, 2009).
[53] HL7 v3 Ballot Site, Universal Domains, Medical Records Domain, Document Query Topic, Interactions, Find Document Metadata and Content Query. (2009, Mar.). [Online]. Available: http://www.hl7.org/v3ballot/html/welcome/environment/index.htm (accessed April 12, 2009).
[54] Java Platform, JDK6 Documentation, Standard Edition 6. (2009, Apr. 12). [Online]. Available: http://java.sun.com/javase/6/docs/api/
[55] Adobe Flex 2.0.1. (2009, Apr. 12). [Online]. Available: http://www.adobe.com/products/flex/
[56] Testing and Test Control Notation (TTCN), European Telecommunications Standard Institute (ETSI). (2009, Apr. 12). [Online]. Available: http://www.ttcn-3.org
[57] European Telecommunications Standard Institute (ETSI), France. (2009, Apr. 12). [Online]. Available: http://www.etsi.net/WebSite/homepage.aspx
[58] OASIS ebXML Test Framework v1.0. (2004, Oct.). [Online]. Available: http://www.oasis-open.org/committees/download.php/10896/IIC_ebXMLTestFramework_v1.1_10_11_04_final_draft.zip (accessed April 12, 2009).
[59] NIST Manufacturing Business to Business (B2B) Interoperability Testbed. (2009, Apr. 12). [Online]. Available: http://www.mel.nist.gov/msid/b2btestbed/
[60] National Institute of Standards and Technology. (2009, Apr. 12). [Online]. Available: http://www.nist.gov/
[61] Standards for Technology in Automotive Retail. (2009, Apr.). [Online]. Available: http://www.starstandard.org/ (accessed April 12, 2009).
[62] HL7 Message Maker. (2006, Jan.). [Online]. Available: http://www.itl.nist.gov/div897/ctg/messagemaker/ (accessed April 12, 2009).
[63] Integrated Healthcare Enterprise (IHE), Chicago IL. (2009, Apr. 12). [Online]. Available: http://www.ihe.net/
[64] IHE Test Tools. (2009, Apr.). [Online]. Available: http://ihedoc.wustl.edu/mesasoftware/index.htm
[65] IHE Cross Enterprise Document Sharing (XDS) Profile. (2007, Aug.). [Online]. Available: http://static.ihe.net/Technical_Framework/upload/IHE_ITI_TF_4_0_Vol1_FT_2007_08_22.pdf (accessed April 12, 2009).
[66] IHE Basic Patient Privacy Consent (BPPC) Profile. (2007, Aug.). [Online]. Available: http://static.ihe.net/Technical_Framework/upload/IHE_PCC_TF_Vol_1_TI_2007_08_15.pdf (accessed April 12, 2009).

Tuncay Namli is currently working toward the Ph.D. degree from the Department of Computer Engineering, Middle East Technical University (METU), Ankara, Turkey.

He is currently a Senior Researcher at Software R&D Center (SRDC), METU. His current research interests include semantic interoperability, Web service technology, and health care informatics.

Gunes Aluc is currently working toward the M.Sc. degree from the Department of Computer Engineering, Middle East Technical University (METU), Ankara, Turkey.

He is currently a Senior Researcher at Software R&D Center (SRDC), METU. His current research interests include conformance and interoperability testing, semantic interoperability, Web service technology, and health care informatics.

Asuman Dogac received the Ph.D. degree in computer engineering from the Middle East Technical University, Ankara, Turkey, in 1980.

She is currently a Full Professor at the Department of Computer Engineering, Middle East Technical University (METU), Ankara, Turkey, and the General Manager of the Software R&D Center (SRDC), METU. Her current research interests include health care informatics, Web services, and semantic interoperability.