Towards Semantic Healthcare with Interoperable Processes

By
Wajahat Ali Khan
2007-NUST-MS-PhD IT-16

Supervisor
Dr. Khalid Latif
Department of Computing

A thesis submitted in partial fulfillment of the requirements for the degree of Masters of Science in Information Technology (MS IT)

In
School of Electrical Engineering and Computer Science,
National University of Sciences and Technology (NUST),
Islamabad, Pakistan.

(July 2010)

Abstract

Healthcare systems face severe issues in interoperability because of gaps in inter processes communication. Researchers and practitioner are trying to make systems interoperable and integrate for the benefit of all the stakeholders including hospitals, clinicians, medical support staff, and patients. Interoperability, however, can only be achieved when standards are practiced. HL7 is one of the key standard for communication of medical information across and within the healthcare systems. Two different healthcare systems can earn HL7 conformance and compliance, yet can be incompatible for integration because of varying implementation of HL7 interaction model. This is mainly because workflows in healthcare systems are very complex. Their interoperability on one hand requires flexible mechanism for the mapping of business processes to a standard. On the other hand, it requires deeper understanding of the standard HL7 interaction model and cracks created by their incompatible implementations. In this thesis a novel approach is proposed for dynamically creating semantic web services as overlay on top of the existing services. These semantic services are mapped to the interaction model ontology. Integrated reasoning mechanism in WSMX framework provides necessary execution semantics for more effective and seamless end-to-end communication. The proposed solution complements the existing semantic data interoperability in HL7 and leads to true semantic process interoperability.

Certificate of Originality

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person nor material which to a substantial extent has been accepted for the award of any degree or diploma at SEECS or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at SEECS or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics which has been acknowledged.

Author Name: **Wajahat Ali Khan**

Signature: ________________

Acknowledgments

First and foremost, I am immensely thankful to Almighty Allah for letting me pursue and fulfill my dreams. Nothing could have been possible without His blessings.
I am deeply indebted to my supervisor, Dr Khalid Latif, for providing me the supervision, motivation and encouragement throughout the span of my work. His insight, breadth of knowledge, and enthusiasm has been invaluable. Without his care and able guidance, I would not have been able to complete my thesis.
I am grateful to my co-supervisor, Dr Hafiz Farooq Ahmad, for his valuable suggestions and guidance in my research. He has been abundantly helpful, and I owe a lot to him.
I am thankful to National University of Science and Technology (NUST), Pakistan for providing me scholarship due to which I carried out my research activities with good mind set. I would like to thank my team leads, Mr. Maqbool Hussain and Mr. Muhammad Afzal and all HLH project members for their valuable suggestions and for the many enjoyable discussions we had.
I am also whole heartedly thankful to Mr Omair Shafiq, Mr Shaban Jokhio, PhD Student at University of Auckland, New Zealand and Mr Federico Fecca, WSMX Technical Team Member for their support and guidance that took me out of some very critical situations in my implementation phases of thesis.

Wajahat Ali Khan

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction and Motivation

1.1 Introduction

According to a report \$ 86.6 billion can be saved annually, if standardized information exchange takes place in healthcare applications [27]. The use of Information and Communication Technology (ICT) promises to facilitate healthcare in information sharing and exchange, collaboration between healthcare stakeholders, and transparency [17]. The challenge of interoperability is faced by the world due to variation in implementation of data and processes. Variation in the standards have also lead to the interoperability issues. These issues can only be resolved when standards are practiced. In this thesis one of the leading healthcare standard is discussed, known as Health Level Seven (HL7)[1]. More specifically, this thesis focuses on bringing semantics in information exchange based on interactions supported by HL7 V3 specifications.

[1]http://www.hl7.org/

1.2 Health Level Seven (HL7) and Semantic Interoperability

HL7 is a messaging standard that is used for the exchange of medical information between different communicating parties or systems. It also helps in the delivery and evaluation of health services. It helps in the exchange, integration, sharing and retrieval of electronic health information [4]. The most commonly used versions of HL7 are HL7 V2.x and HL7 V3. HL7 V 2.x is mainly focused on the transfer of message from sender to the receiver rather on interoperability. HL7 V3 focused on the shortcomings of HL7 V2.x and overcome those by targeting semantic interoperability. HL7 V3 is based on the standard model called Reference Information Model (RIM) [9].

A message is designed in HL7 V3 using four core models: use case model, information model, interaction model and message design model. These models define the expected behavior of the message to be sent. Use case model provides the modeling tools to develop the requirements for HL7 messages [10]. Information model provides the modeling tools to define the information used in HL7 messages. It also discusses the principles behind the HL7 Reference Information Model (RIM). Interaction model provides the modeling tools for defining the interactions, trigger events and application roles. The message design model depends on use case model, information model and interaction model for defining the format of HL7 message. These models are the backbone of the semantic interoperability support in HL7.

Semantic interoperability is the way to intelligently interpret the transferred knowledge among communicating machines and provide accurate desired results. HL7 V3 provides specifications for different domains like patient administration, specimen, laboratory, observation etc. Every domain supports data and processes particular to that domain in addition to some common elements that are shared among multiple domains. The main focus of this thesis is to bring semantics in the process communication related to the interactions included in laboratory domain. This work provides a new perspective on semantic interoperability by representing the interactions and other artifacts with ontologies.

One technique for achieving semantic process interoperability is to use web services. "Web services provide a standard means of interoperating between different software applications, running on a variety of platforms and/or frameworks" [6]. The complexity is increased for web services when semantic and syntactic heterogeneities are brought in to consideration for the transfer of messages between systems. Therefore, there is a need of using semantic web services for achieving semantic process interoperability. Semantic web services can be used for enhancing the web services capabilities in understanding semantics such that it can be more easily machine processable. This will result in better machine understanding of the web services and the communication would be more effective. Semantic web services should have proper pre-condition, post-condition, effects and assumptions. There are different approaches used for realizing semantic web services. These include Semantic Annotations for Web Service Description Language (SAWSDL) [19], Semantic Annotation for Representational State Transfer (SA-REST) [20], Ontology Web Language Service (OWL-S) [26], Semantic Web Services Framework (SWSF) [8] and Web Service Modeling Ontology (WSMO) [15]. WSMO is used in this thesis as it is the most effective and complete approach amongst all.

In order to achieve semantic interoperability, Web Service Modeling Framework (WSMF) and Health Life Horizon (HLH)[2] architectures are to be integrated. **Health Life Horizon (HLH)** is a project related to healthcare and has provided the framework for interoperable messaging system between medical sys-

[2]http://hl7.seecs.nust.edu.pk/

tems. The proposed system is also part of the HLH project and it is helping in handling the processes of HL7 V3. HLH architecture is composed of three main components HLH database mapper component used for the generation of mapping specification. This mapping specification file is created by the mapping of database with the RIM model and it is then used by HLH Core Engine Component. HLH Core Engine component is used for HL7 message generation and parsing and also handling semantic with its Ontology Core component. This component then forwards the message to the HLH transportation component which is used for communicating message from sender to the receiver. Different protocols like Minimal Lower Layer Protocol (MLLP), Web Services and ebXML can be used for message exchange [7].

1.3 Motivation

The two most important issues that the healthcare industry is facing are integration and interoperability of systems. The broader goal of interoperability can only be achieved when standards are practiced. Therefore there is requirement of standards that provide semantic interoperability. HL7 V3 claims to provide semantic interoperability. It mainly focuses on the semantic data interoperability and semantic process interoperability is still a grey area. To achieve semantic interoperability there is a need of a framework that can support the required constructs for semantic interoperability. WSMF [18] provides the basis for providing semantics in processes. WSMO which contains following entities: ontologies, mediators, web services and goals, Web Service Modeling Framework (WSML) [14] and Web Service Execution Environment (WSMX) [11].

1.4 Thesis Objective

In HL7 the interoperability can be seen from two perspectives; data and process. The potential benefits of semantic data interoperability remain in doubt without semantic process interoperability. Achieving interoperable data would be less effective if semantics in the communication components are lacking. HL7 V3 provides semantic interoperability using common terminologies while process interoperability is a gray area. Semantic data interoperability means understanding of the data communicated between sender and receiver in such a way that the receiver easily interprets the sender intension of sending the data and properly responds. On the other hand, semantic process interoperability is the type of semantic interoperability, which helps in the decision process of the participating parties in communication of HL7 messages on the basis of data contents intended to be exchanged for automation [24]. For bringing semantic interoperability in the HL7 processes, semantic web services are followed for the communication.

1.5 Main Contributions

This thesis is based on the design and implementation of Semantic Healthcare with Interoperable Processes. The proposed system achieves the objective "semantic process interoperability" by the integration of HL7 and SWS architecture. The main activities include alignment of HL7 artifacts into SWS concepts that ultimately lead to semantic process interoperability. The process related HL7 concepts are aligned into the WSMO entity ontology and Interaction Ontology is modeled. HL7 message contents are mapped to message ontology. The system provides semantic web services representing functionalities in terms of HL7 Application Roles. These semantic web services consists of choreographies which handles the overall process interoperability. The goals are responsible for finding out the related web services and process mediator for handling the choreographies for the overall process execution. The design of choreographies is derived from HL7 storyboards with composition of corresponding HL7 interactions.

In particular, the contribution to this work includes:

- Investigation of the role semantic web services play in HL7 V3 standard processes.
- Design and development of WSMF artifacts mapping with HL7 V3 artifacts.
- HL7 based architecture design for seamless interaction of laboratory services using ontologies.
- Process automation in HL7 based healthcare messaging system by developing semantic web services.

1.6 Overview of Thesis

The rest of the thesis is structured as follows. Chapter 2, provides the details related to models and artifacts of HL7 that are the main building blocks for achieving process interoperability. It highlights the information related to messaging components that are divided in to two parts related: behavior and structure of the message. Chapter 3 moves forward from HL7 to the role of semantic web services in healthcare. The work describes the main components of semantic web service architecture. It explains the top entities of WSMO, the different variants of WSML and the components of WSMX. This chapter also explains the related work to our system describing different healthcare systems using semantic web services. Furthermore

Chapter 4, elaborates the HLH studio and the semantic web services architecture. It also explains the relation of both the architecture components with each other and the flow of information in achieving semantic process interoperability. In order to implement and integrate both architectures Chapter 5, describes the modeling of WSMO entities using WSML and then implementation of the WSMX components. The modeling is explained using the WSMT toolkit and wsmo4j API is also used for implementation. In order to evaluate effectiveness of the system Chapter 6, explains the criteria by which the system developed is evaluated. The system is compared with the current HL7 system without semantics. The semantic system developed has some advantages as compare to the system without semantics. Chapter 7, concludes the research work and revisits contributions. Future work is also explained in the light of contributions of the thesis.

Chapter 2

HL7 V3 Process Artifacts for the Laboratory Domain

This chapter provides insight to HL7 V3 domains and models and their role in message development and process communication. Also the workflows based on process artifacts are discussed.

2.1 HL7 V3 Artifacts

For HL7 V3 message development framework, some phases, activities and models are to be followed as prerequisite steps. These are categorized into two parts: Requirement Analysis and Solution Design and Implementation. Requirement Analysis part consist of Use Case Model and Reference Information Model (RIM). The Solution Design and Implementation phase consists of Interaction Model, Hierarchical Message Description (HMD) and Implementation Technology Specifications (ITS). In order to achieve semantic process interoperability we need to take into account the Interaction Model and Hierarchical Message Description. All these models will lead to the HL7 message development. There are two types of HL7 V3 messaging components: Static and Dynamic. The static components shows the structure of the message where as the dynamic components shows the behavior of the message. The static message components includes DMIM, RMIM, HMD and MT where as the dynamic components includes Application Roles (AR), Trigger Events (TE) and Interactions. HL7 V3 is composed of different models to be followed for achieving semantic interoperability. These models consist of Use Case Model, Information Model, Interaction Model and Message Description Model. Interaction and Information Models are related to the healthcare processes directly. Information model is mainly responsible for handling the information related to HL7 message. Interaction model is related to the process artifacts that are used for handling the overall communication of HL7 V3 [10]. Process interoperability is more related with the dynamic components of the message development, therefore more focus would be on the interaction model in this thesis.

2.2 HL7 Interaction Model

Interaction model is responsible for the communication of messages between communicating devices. Interaction model consists of process artifacts such as: application roles, trigger events, message types and interactions. These artifacts are responsible for the communication of messages between the communicating parties. Application roles are the type of process artifacts that are responsible for sending and receiving of the messages. These can be categorized into two parts: Sending and Receiving Application Roles. Interactions provide the associations between the communicating application roles and these interactions are initiated by another process artifact called as trigger event. Another important type is called Message type that constitutes the set of rules for constructing the message given specific set of instance data. A simple scenario of all the process artifacts is that of sending the test result to a laboratory from a clinical point.

The application role for sending the message is Order Placer of the laboratory domain. Order Fulfiller acts as a receiving AR. The interaction Order Fulfillment Request is triggered by the Order Fulfillment Request trigger event. All these process artifacts are related to the laboratory domain of HL7 V3 [5].

Process artifacts are the components that are responsible for handling the behavioral aspects of HL7 processes. These include application roles, interactions, trigger events and message types. Application Roles are the logical components that are used for the communication of interactions between the sender and receiver. Interactions are the flow of information to be transferred between communicating parties. Message types consist of the information to be carried out in the message. Trigger events are used for the initiation of the interaction. Figure 2.1 shows the process artifacts in HL7 V3 and also their relationship with each other. As mentioned earlier, only the process artifacts of the laboratory domain are considered in the presented case study.

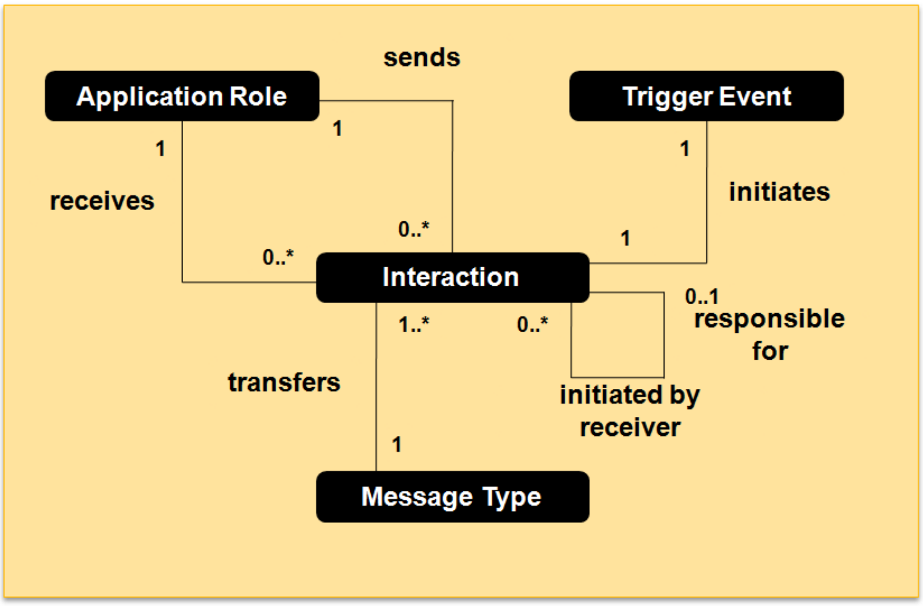

Figure 2.1: Interaction Model

The communication process between communicating parties involves set of process artifacts. The process is initiated by a trigger event for communication of the message. Each interaction is associated with a message type used for the contents of the message. The interaction is carried out by a specific sending application role with receiving application role. The application roles are the logical components that uses set of interactions for communication between sender and receiver. There are three set of interactions in HL7 V3 laboratory domain i.e. Order, Promise and Result. An organization can use any of the interactions from these three categories of interactions in their workflows based on their requirements. The interaction model consists of the process artifacts that are responsible for the transfer of HL7 message from sender to the receiver. It provides all the messaging requirements and also links both Use Case model, Information model with message definition process [10]. The trigger event in the interaction model that causes the flow of information between the participating roles in the information exchange. All these process artifacts show the behavior of the HL7 message development process. An application role can act as a sender or receiver for communicating message. Interactions uniquely specify associations between trigger events, receiver responsibilities and message types. They also enlist the sending and receiving AR that participate in the process. Trigger event initiates the interaction and a message type specifies the data and its order of appearance in a message.

2.3 Semantic Process Interoperability and HL7

Healthcare workflows are more complicated than industrial workflows as they are non-linear, multi directional, interrupt driven and have unlimited complexity [28]. Healthcare workflows can be classified in to different categories including Administrative (related to patients and healthcare organizations), Financial (manage the finance of healthcare organization), Clinical (manages the operational, decisional and therapeutic decisions related to clinic) and Laboratory (manages the data for diagnosis) [31]. For brevity we will only discuss laboratory domain workflows for healthcare organizations that are HL7 V3 compliant.

To make healthcare systems using HL7 standard semantically interoperable, there is a need to make use of the specifications of the HL7 processes and also the role of messages in it. Any two organizations should be able to communicate autonomously even if their workflows are not homogeneous.

Consider, for example, two organizations with different set of interactions as shown in Figure 2.2. Source Organization supports interactions related to all the three categories supported in laboratory domain of HL7 V3 i.e. order, promise and result. Receiver Organization only supports order and result categories of interactions. The two application roles (AR) that are involved in performing these interactions are Order Placer (OP) and Order Filler (OF). The steps in the process to handle heterogeneity are as follows.

Step 1: Source Organization interacts with the receiver organization using Order Fulfillment Request interaction communicated by Order Placer AR to Order Fulfiller AR. Message Type (MT) Placer Order is associated with Order Fulfillment Request interaction and Order Activate trigger event (TE) is used for initiating this interaction.

Step 2: Source Organization is expecting Promise Activate interaction from the Order Fulfiller AR of the receiver organization to the Order Placer AR of the source organization. It is expecting this interaction because it is part of the workflow of this organization while in workflow of receiver organization; the response for Order Fulfillment Request is Order Confirm. Blockade will take place with the system having no semantics.

Therefore no further communication will take place until the deadlock is resolved.

Step 3: Semantics can help the system to resolve the deadlock by matching Order Confirm interaction in source organization workflow. Since interaction exists in its workflow therefore it accepts this interaction and waits for Result Complete with Fulfillment interaction.

Step 4: Receiver Organization interacts with source organization using interaction Result Complete with Fulfillment as described in its workflow. It uses the MT and TE of Result Event and Result Complete with Fulfillment respectively.

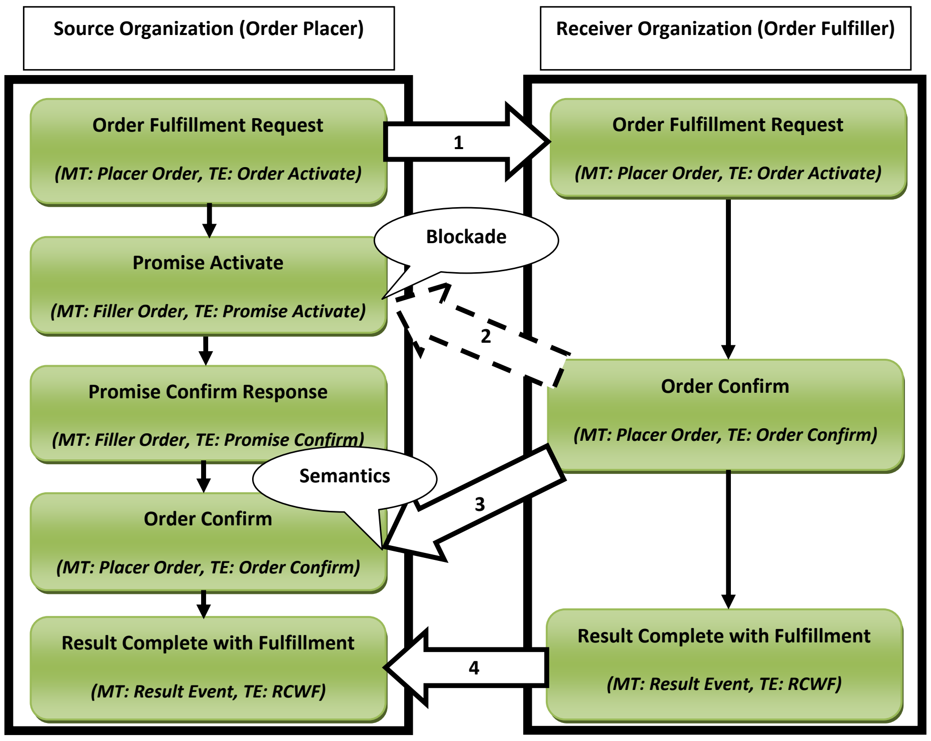
Source Organization (Order Placer)
Receiver Organization (Order Fulfiller)
Order Fulfillment Request
(MT: Placer Order, TE: Order Activate)
1
Order Fulfillment Request
(MT: Placer Order, TE: Order Activate)
Blockade
Promise Activate
(MT: Filler Order, TE: Promise Activate)
2
Order Confirm
(MT: Placer Order, TE: Order Confirm)
Promise Confirm Response
(MT: Filler Order, TE: Promise Confirm)
Semantics
3
Order Confirm
(MT: Placer Order, TE: Order Confirm)
Result Complete with Fulfillment
(MT: Result Event, TE: RCWF)
4
Result Complete with Fulfillment
(MT: Result Event, TE: RCWF)

Figure 2.2: Workflow Heterogeneity in Healthcare Systems

Semantic web technologies provides us the platform for making HL7 completely semantic interoperable. The process artifacts in HL7 V3 can be handled by using semantic web technologies and with the help of semantic web services. “Semantic Web Services aim to reduce the human effort required to build Service Oriented Architectures by enabling machines to understand the function and interfaces of Web services through the addition of semantics.” [23].

The process artifacts information of HL7 V3 should be incorporated using WSMO and thus machines should be made intelligent to handle process interoperability. The proposed system is the system that will achieve semantic process interoperability by integrating WSMF and HLH architectures. It is based on semantic web services in the form of application roles and understandability to the system related to process is provided by ontologies (Interaction and Transmission in the proposed system).

2.4 Laboratory Domain Process Artifacts

As stated earlier, this thesis is related with only the laboratory domain of HL7 V3. Specifications published during January 2007 [5] are used and referred throughout this thesis. Laboratory domain process artifacts are as follows:

2.4.1 Application Roles

Application roles are abstractions that standardize the roles played by healthcare information system components when they send or receive a message. The application roles for the laboratory domain are as follows:

Order Placer

This AR is responsible for changes in the state of the order and it places an order from Order Fulfiller AR.

Order Fulfiller

This AR is responsible for managing the state of promises and events in response to changes in the state of the orders, also it send observation results and receives orders.

Result Receiver

This AR is responsible for receiving a message from another system. Being a tracker application it is also responsible for state change of a laboratory observation.

Result Query Placer

This AR is responsible for sending originating the queries for results and also is capable of receiving query response.

Result Query Filler

This AR satisfies the request related to result queries from the result query placer AR.

2.4.2 Trigger Events

Find Result

This TE is related to the information of a result requested.

Find Result Response

This TE is related to information regarding a result that has been found in response to a request.

Result Abort

This TE is related to the result event that has been aborted. There would be no additional work performed for the associated testing.

Result Activate

This TE is initiated when the result is reported for the first time.

Result Complete

The result event has been aborted and no additional work will be done performing the associated testing.

Result Complete with Fulfillment

This TE is initiated when the completion of the laboratory observation takes place and the result is complete from all aspects that were demanded initially.

Result Confirm

This TE is initiated when the result interactions are confirmed that they are received.

Result Corrected

When some changes are carried out in a result that was completed but revisited due to some changes demanded, result corrected TE is initiated.

Result in Progress

When the preliminary results are available and the actual result observations are being reported then this TE is initiated.

Result Nullify

This TE is initiated when mistakenly the result was created and should be removed completely.

Result Reject

This TE is used when the result should be rejected due to some conditions that the result doesn't fulfill.

Result Status

This TE indicates some changes occurred in the status of the result.

2.4.3 Interactions

Result Status

This interaction is used when the changes are required to be made in the result and the process of at which stage the result resides.

Result Reject

This interaction is used when the result should be rejected due to some conditions that the result doesn't fulfill.

Result Nullify

This interaction is initiated when mistakenly the result was created and should be removed completely.

Result in Progress

When the preliminary results are available and the actual result observations are being reported then this interaction is carried out between the AR's.

Result Corrected

When some changes are carried out in a result that was completed but revisited due to some changes demanded, result corrected interaction takes place between sending and receiving AR's.

Result Confirm

This interaction is used for the confirmation of a receipt of a result.

Result Complete with Fulfillment

This interaction takes place when the state of the result is complete and final.

Result Complete

This interaction indicates that the report of the result is complete and the result is near to fulfillment.

Result Activate

When the preliminary report is not available but the laboratory has information to be communicated, result activate interaction is carried out.

Result Abort

This interaction indicates that the result has not completed and is stopped before it has completed.

Find Result Query Response

This interaction is related to information regarding a result that has been found in response to a request.

Find Result Query

This interaction is related to the information of a result requested. Due to this interaction the query related to a message is requested from the concerned receiver.

2.4.4 Message Types

The message types are used to communicate queries for observations. These are part of HMD's.

Result Query

These can be as simple as queries for single test, order for a battery of tests, or more complex groupings of tests such as microbiology observations.

Result Event

This MT is used for the communication of laboratory observations and is based on HMD. The tests in it include simple and complex test.

Minimal Event Act Reference

This artifact is used to identify an act.

These process artifacts are used for modeling ontologies to be used for bringing semantics in the system.

2.5 Interaction Workflow Description

2.5.1 HL7 Laboratory Domain Interactions

Application roles (AR) are the components used for the communication of message from sender side to the receiver side. These application roles communicates message in the form of interactions. The interactions are initiated by trigger events. The information in this document is taken from the HL7 V3 ballot January 2007 [5]. The interactions are divided into three topics

- Placer Topic
- Filler Topic
- Result Topic

The placer and filler topics are only related with the communication between the application roles Order Placer and Order Filler. While the result topic consist of interactions that has the responsibility of communicating the message between application roles such as Order Placer, Order Fulfiller, Result Query Placer, Result Query Fulfiller and Result Receiver.

Interactions in Placer Topic

The interactions in placer topic are only communicated by the application role Order Placer. These interactions are

- Order Cancel
- Order Fulfillment Request
- Order Nullify
- Order Replace
- Order Revision
- Promise Confirm Response
- Promise Reject

Interactions in Fulfiller Topic

The interactions in the fulfiller topic are only communicated by the Order Fulfiller application role. These interactions are

- Order Confirm
- Order Reject
- Promise Cancel
- Promise Status
- Promise Activate
- Promise Fulfillment
- Promise Nullify
- Promise Replace
- Promise Revision
- Promise Status Change

Interactions in Result Topic

All the application roles that are in the laboratory domain have interactions used in some way in the result topic. The interactions used in this document are taken from the universal and global domain in the ballot related to the laboratory domain. Combine interactions from both the domains are used in this workflow. The interactions are

- Result Complete
- Result Confirm
- Result Corrected

- Result in Progress
- Result Nullify
- Result Reject
- Result Replace
- Result State Change
- Result Status
- Find Result Query
- Find Result Query Response
- Find Activate
- Result Complete with Fulfillment

A comprehensive workflow is provided in Figure 2.3. Two healthcare organizations can only communicate provided that they are, first and far most important, HL7 compliant, and that they understand the intricacies and heterogeneity in the workflows. It is not compulsory for organizations to have all the three sets of interactions in their workflows for laboratory domain. Some organizations will follow all the three sets of interactions while others would only prefer using two sets of interactions in their workflows leading to heterogeneities in workflows. Therefore a system is required to handle these heterogeneities. The proposed system is based on resolving these issues of heterogeneities in HL7 processes.

2.5.2 Workflow of Interactions in the Laboratory Domain

When the sender wants to communicate the message initially with the receiver the Order Placer and Order Fulfiller application roles will participate in the interaction.

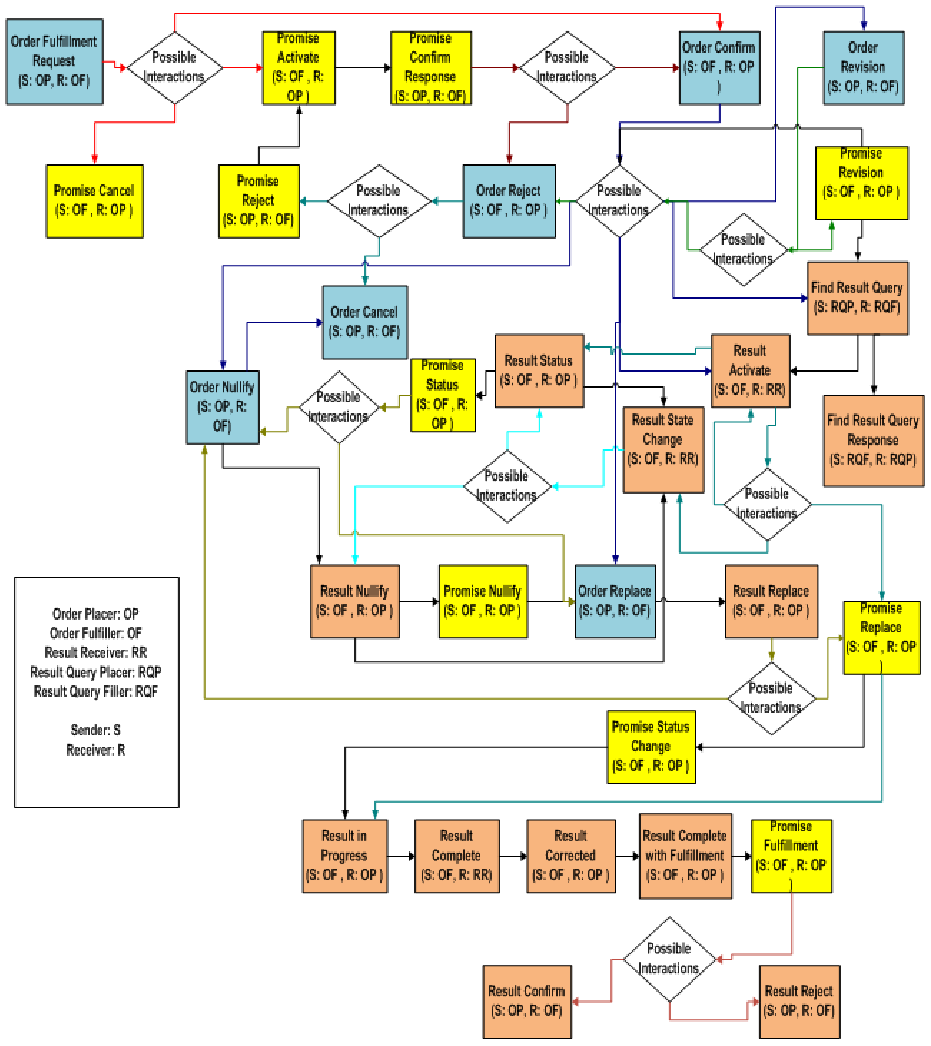
Order Fulfillment Request (S: OP, R: OF)
Possible Interactions
Promise Activate (S: OF , R: OP)
Promise Confirm Response (S: OP, R: OF)
Possible Interactions
Order Confirm (S: OF , R: OP)
Order Revision (S: OP, R: OF)
Promise Cancel (S: OF , R: OP)
Promise Reject (S: OP, R: OF)
Possible Interactions
Order Reject (S: OF , R: OP)
Possible Interactions
Promise Revision (S: OF , R: OP)
Possible Interactions
Find Result Query (S: RQP, R: RQF)
Order Cancel (S: OP, R: OF)
Result Status (S: OF , R: OP)
Result Activate (S: OF, R: RR)
Order Nullify (S: OP, R: OF)
Possible Interactions
Promise Status (S: OF , R: OP)
Result State Change (S: OF, R: RR)
Find Result Query Response (S: RQF, R: RQP)
Possible Interactions
Possible Interactions
Result Nullify (S: OF , R: OP)
Promise Nullify (S: OF , R: OP)
Order Replace (S: OP, R: OF)
Result Replace (S: OF , R: OP)
Promise Replace (S: OF , R: OP
Order Placer: OP
Order Fulfiller: OF
Result Receiver: RR
Result Query Placer: RQP
Result Query Filler: RQF
Sender: S
Receiver: R
Possible Interactions
Promise Status Change (S: OF , R: OP)
Result in Progress (S: OF , R: OP)
Result Complete (S: OF, R: RR)
Result Corrected (S: OF , R: OP)
Result Complete with Fulfillment (S: OF , R: OP)
Promise Fulfillment (S: OF , R: OP
Possible Interactions
Result Confirm (S: OP, R: OF)
Result Reject (S: OP, R: OF)

Figure 2.3: Laboratory Domain Interactions Workflow

- The initial interaction is the **Order Fulfillment Request**, communicated by the Order Placer AR with the Order Fulfiller AR. The Order Placer AR behaves as a sender and Order Fulfiller behaves as a receiver.
- There are three possibilities that the Order Fulfiller AR can give in response to the Order Fulfillment Request.
  - If the Order Fulfiller AR is sure of accomplishing the

  Order Fulfillment Request, it can give response in the form of **Promise Activate** interaction.

  - Else if the Order Fulfiller AR cannot fulfill the requirements mentioned in the Order Fulfillment Request, the response would be **Promise Cancel** interaction. Promise Cancel would result in the end of the interactions.
  - The third response that the Order Fulfiller AR can give is **Order Confirm**. It can also happen that the Order Fulfiller directly confirm the order rather than first doing Promise Activate interaction.

- If the response of the Order Fulfiller AR is Promise Activate AR, then further interaction of the Order Placer AR will be **Promise Confirm Response**. This interaction is between the Order Placer AR and Order Fulfiller AR, as sender and receiver respectively.

- The Promise Confirm Response interaction is communicated with the Order Fulfiller AR by the Order Placer AR. The response of this interaction has two possible interactions that the Order Fulfiller AR can send.

  - As previously described, if the Order Fulfiller AR wants to go ahead with the fulfillment of interaction, the interaction would be **Order Confirm**.
  - If the Order Fulfiller AR is not sure about the fulfillment of the order, then **Order Reject** interaction is send to the Order Placer AR. There are two possible interactions that the Order Placer AR can interact with the Order Fulfiller AR.
    - Order Placer AR can send the **Promise Reject** interaction; if it accepts the Order Reject interaction

of the Order Fulfiller AR. It means that the interactions about the order can start from the Promise Activate interaction again.

    * The Order Placer AR can send the **Order Cancel** interaction, to completely finish further interactions about the order.

- The possible interactions after the Order Confirm interaction that the Order Placer or Order Receiver can communicate are:
  - **Order Revision** interaction by the Order Placer AR, if there are some revisions required by the order originator like change in some specimens required to be communicated to the Order Fulfiller AR.
    * The Order Fulfiller AR can send the **Promise Revise** interaction, if it can fulfill the revised requirements else would send **Order Reject** interaction. The promise revise interaction would lead to the **Result Activate** interaction explained earlier.
  - **Order Cancel** interaction by the Order Placer AR if the message order is no longer required. The Order Placer AR feels that the interaction is no more required.
  - **Find Result Query** interaction by the Result Query Placer AR for some query about the result. This interaction is communicated to the Result Query Fulfiller AR.
    * The response to the Find Result Query interaction would be **Find Result Query Response** interaction by the Result Query Fulfiller AR to the Result Query Placer AR.

  - **Result Activate** interaction is send by the Order Fulfiller AR to the Result Receiver AR in order to start processing on the result.
  - **Order Replace** interaction can be send by the Order Placer AR to the Order Fulfiller AR. This should be communicated because the order has to be replaced.
  - **Order Cancel** interaction can also take place, if the order is no more required.

- There is also number of possible interactions after the Result Activate interaction by the Order Fulfiller AR to the Result Receiver AR. The Result Receiver AR is used for keeping the track of different result activities.

  - **Result State Change** interaction can take place, for example the result can change the state to the complete state or abort state. Therefore the Result State Change interaction should be communicated with the Result Receiver AR by the Order Fulfiller AR. After this interaction there are two possible interactions that can take place. Both of these interactions are sent by Order Fulfiller AR to the Order Placer AR.
    - **Result Status** interaction explained later can be send after the Result State Change interaction.
    - **Result Nullify** interaction can also be send after the Result State Change interaction. This interaction is send by the Order Fulfiller AR to the Order Placer AR to convey the message that the result cannot be further performed for the time and can be performed later on but not necessarily.
  - **Result in Progress** interaction can also be send by the Order Fulfiller AR to the Order Placer, in order to inform it about the current status of the result.

  - **Result Status** interaction can also be send after the Result Activate, showing the status of the result. The difference between Result Status and Result State Change interaction is that the former should be sent after some specific time e.g. after three specimens are prepared and the later one can be communicated anytime. A **Result State Change** interaction can occur after this as well.

- The next interaction after the Result Status interaction is **Promise Status** interaction which is sent by the Order Fulfiller AR to the Order Placer AR. This is an optional interaction to be carried out by the Order Fulfiller AR. After the Result Status or Promise Status interactions there are two possible interactions that can take place:

  - **Order Replace** interaction can occur after the Result Status or Promise Status interaction.
  - **Order Nullify** interaction can be send by the Order Placer AR to the Order Fulfiller AR. This interaction means that the order has been cancelled for the time being and can be performed later on. There is a difference between Order Cancel interaction and Order Nullify interaction. The Order Cancel interaction occurs before Result Activate interaction such that no process about the result has started and it also means that the order has completely finished and should not start again. On the other hand the Order Nullify interaction is send after the result has been activated if required and the order is suspended for the time being and can start later on.

- **Result Nullify** interaction is send by the Order Fulfiller AR after the Order Nullify interaction.The difference between Order Nullify and Result Nullify is that Order Nullify is send by the Order Placer AR whereas the Result Nullify is send by the Order Fulfiller AR. After this interaction the Result State Change interaction can also be send to the Result Receiver AR for keeping track of the result. So Order Nullify is from the order generator side and Result Nullify is from the result generator side.

- **Promise Nullify** interaction is send by the Order Fulfiller AR to the Order Placer AR. This interaction is send to convey the message of the Order Fulfiller AR about unable to fulfill the promise. This is also an optional interaction after the Result Nullify interaction.
- **Order Replace** interaction can be send by Order Placer AR to the Order Fulfiller AR for replacement of the order as explained earlier.
- **Result Replace** interaction is send by the Order Fulfiller AR to the Order Placer AR after the Order Replace interaction has been received. This is to inform Order Placer AR that the replacement has been carried out. There are two possible interactions after this interaction.
  - The Order Placer AR after receiving the result replace interaction should not be satisfied from the output therefore should convey **Order Nullify** interaction to the Order Fulfiller AR.
  - **Promise Replace** interaction can be send by the Order Fulfiller AR to the Order Placer AR to communicate the changes that have taken place to the promise that it has made for fulfilling the order.
- **Promise Status Change** interaction is send after the Promise Replace interaction is communicated to the Or-

der Placer. This interaction is send by the Order Fulfiller AR to inform the Order Placer AR about the current status of the promise.

- **Result in Progress** interaction is forwarded by the Order Fulfiller AR to the result receiver AR for keeping track of the result progress.
- **Result Complete** interaction is send by the Order Fulfiller AR to the Result Receiver AR. The Result Receiver AR stores the complete status of the result with it.
- **Result Corrected** interaction is send by the Order Fulfiller AR to the Order Placer AR for informing it about the completion of the result. The result complete interaction is carried out with the Result Receiver AR and the Result Corrected interaction is carried out with the Order Placer AR.
- **Result complete with Fulfillment** interaction is send by the Order Fulfiller AR to the Order Placer AR to confirm that the order fulfillment has taken place. There are two possible interactions that can be communicated by the Order Placer AR in response to the Result Complete with Fulfillment interaction.
  - **Result Confirm** interaction from the Order Placer AR to the Order Fulfiller AR means that the fulfillment of the promise has been satisfied and the result is complete and correct.
  - **Result Reject** interaction from the Order Placer AR to the Result Receiver AR suggests that the result is not acceptable and the promise has not been fulfilled.

Chapter 3

Semantic Web Services in Healthcare

This chapter describes the role of semantic web services in the healthcare domain. The different approaches used are explained and the most appropriate one is selected for our proposed system implementation. Also the various projects based on semantic web services are highlighted and their critical analysis is discussed.

3.1 Approaches to Semantic Web Services

Web Services are making applications communicate with each other and thus helps in reduced time and cost, related to web applications. Semantics added with web services can lead to automation in the service discovery and composition. There are lots of techniques proposed for semantic web services but the leading amongst them are IRS, OWL S and WSMF [25].

Ontology Web Language Services (OWL-S) is web service framework that uses set of ontologies for description and reasoning of services. The ontologies involved in the service description and reasoning are profile, process model and grounding. The main goal of OWL-S is to provide automation in the discovery, invocation, composition, and interaction with the web service [12, 29].

Web Service Modeling Framework (WSMF) is a semantic web service based approach that works on two main principles strong decoupling and strong mediation service. It is based on two frameworks Semantic Web enabled Web Services (SWWS) [2] and Web Service Modeling Ontology (WSMO). It focuses on bringing automation in the process by automatically discovering and composition of the web services. This framework is further elaborated in Section 3.2.

Internet Reasoning Service (IRS) [1] is a semantic web service based framework used by applications to bring semantics for the description and execution of web services. The different components of IRS II are server, publisher and client. These components communicate through SOAP protocol. The underlying framework on which IRS II is based on is Unified Problem Solving Method Development Language (UPML) that is used for storing knowledge level description [12]. The latest version of IRS II is IRS III that is based on WSMO specifications. It mainly uses the WSMO orchestration aspects for bringing automatic discovery and composition [21].

3.2 Web Service Modeling Framework

WSMF is a framework based on Semantic Service Oriented Architecture (S-SOA). It results in the automation of process and is composed of three sub-architectures including Web Service Modeling Ontology (WSMO), Web Service Modeling Language (WSML) and Web Service Execution Language (WSMX) as shown in Figure 3.1.

[1]http://technologies.kmi.open.ac.uk/irs/

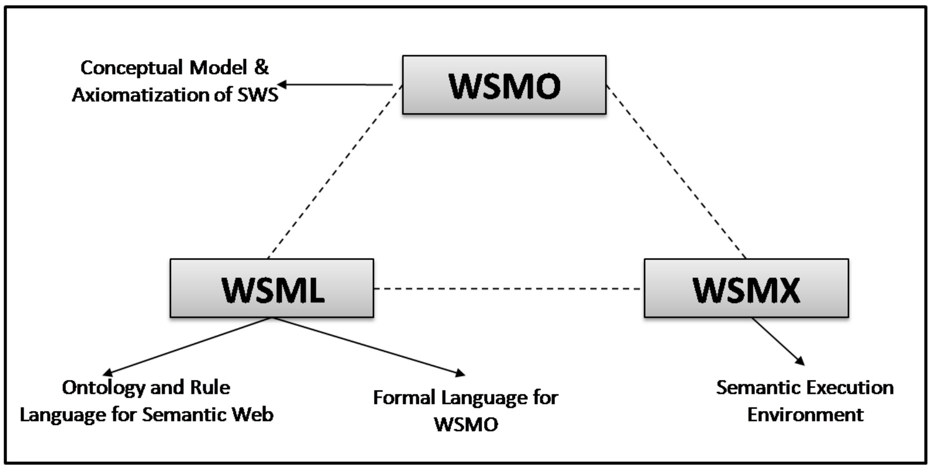

Figure 3.1: Web Service Modeling Framework
[13]

3.2.1 Web Service Modeling Ontology (WSMO)

Web Service Modeling Ontology (WSMO)[2] is a conceptual framework that is developed by ESSI WSMO working group. The top level entities of WSMO are ontologies, web services, mediators and goals. The objective is to use all these WSMO constructs in the HL7 domain. The Web Service Modeling Toolkit is used for the modeling of HL7 ontologies, web services, goals and mediators. Web Service Execution Environment (WSMX)[3] is reference implementation of WSMO and provides execution environment for business application integration. Web Service Modeling Language (WSML)[4] is used for the modeling of WSMO entities by providing different language variants. These variants help in describing the semantic web services. The modeling is done using the toolkit called Web Service Modeling Toolkit (WSMT) [22]. WSMO entities are explained further in the subsequent sections:

[2]http://www.wsmo.org/
[3]http://www.wsmx.org/
[4]http://www.w3.org/2004/12/rules-ws/paper/44/

Ontology

It is the formally specified terminology of information used by all other components. Ontologies are formed by following the specifications like non functional properties, imported ontologies, concepts, axioms, attributes, relations, functions and instances. Ontologies are used for describing semantic web services terminologies and provide the base for them. Mainly Web Service Description Language (WSDL) provides the base for creation of ontology related to certain area. Semantics are added to the web services with the help of ontologies in order to automate the procedure of invocation of a web service.

Web Service

Web services have mainly two parts: the capability and interface information. The capability part of semantic web service consists of the preconditions, post conditions, effects and assumption. The interface portion is based on choreography and orchestration. The choreography is then composed of state signature and transition rules.

Goal

It helps in achieving the objectives that the client desires from the web services. Goal also consists of the capability and interfaces. This builds its relationship with the target web services to be discovered later on.

Mediator

WSMO components are connected with the help of mediator that is used in resolving their heterogeneities. There are four types of mediators used: wwMediator, ooMediator, ggMediator and wgMediator. The wwMediator is used for resolving resolves heterogeneities between web services. The ooMediator is used for resolving the mismatches in terminologies. The ggMediator is use for resolving mismatches between goals to create relationships among them. The wgMediator is used for resolving heterogeneities be web service and goal to help in the discovery of web services.

3.2.2 Web Service Modeling Language (WSML)

Web Service Modeling Language (WSML) is the language used for modeling the WSMO entities. WSML has different variants that are used for modeling purpose and these are based on the logic specification and expressive power as shown in Table 3.1.

Table 3.1: WSML Variants

Variants	**Expressive Power**
WSML-Core	Interaction of Description Logics and Horn Logic (SHIF (D))
WSML-DL	Extends WSML Core to a more expressive DL (SHOIN)
WSML-Rule	Extends WSML Core with Logic Programming primitives
WSML-Full	Unifies WSML-DL & WSML-Rule

3.2.3 Web Service Execution Environment (WSMX)

Web Service Execution Environment (WSMX) provides the execution mechanics of how the web service composition, discovery, ranking, selection and invocation will take place. The message in the form of goal is provided to the WSMX and then on the base of this goal the components of WSMX are executed for achieving the desire of the client. WSMX consists of different components that are shown in Figure 3.2

Discovery

This component of WSMX is used for the discovery of the usable web services that is desired by the client with the help of a goal.

Ranking

The web services discovered will then be ranked on the basis of matching. The ranking is based on the exact match, subsumption etc.

Composition

This component is used to combine services to achieve a goal. A goal can be achieved when the functionality of different services are combined for the desired objective.

Selection

On the basis of ranking component, the exact matched web service is selected for further processing. This service is then forwarded to the invocation component for the end point web service to be invoked.

Mediation

In order to resolve the heterogeneities problems related to the system, mediation component is used. It is used to resolves mismatches (data, protocol, process) that are hampering in interoperation.

Choreography

This component is used to handle the interactions and processes between the service providers and clients. It includes the state signature and the transition rules to handle the flow of information.

Invocation

This component is used for invocation of the end point web service deployed at a web server. It is based on the information that is provided in the choreography of the semantic web service.

Grounding

WSMX only understands WSML language while the messages are in the form of XML, therefore a mechanism is required for conversion between both forms. This is handled by the grounding component which is responsible for lifting and lowering between the semantic and syntactic data representations.

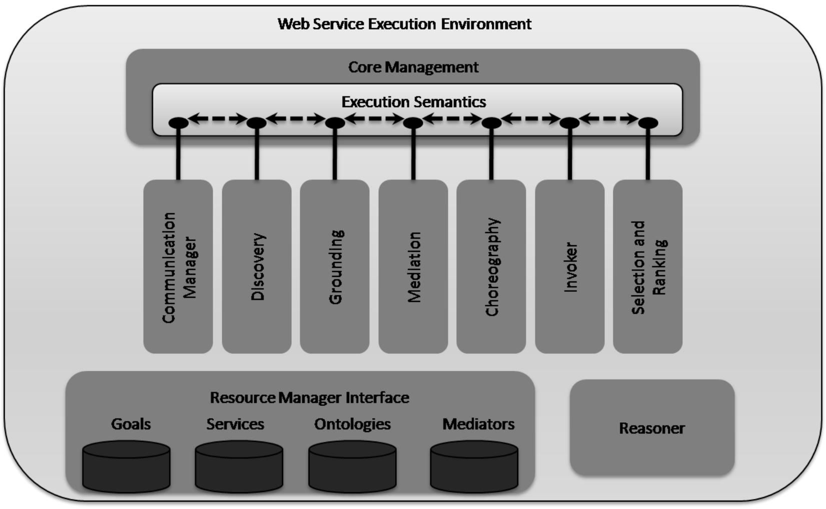

Figure 3.2: Web Service Execution Environment

WSMF framework is based on these three sub architectures and results in bringing the automation of the process. There are different tools available for modeling of the WSMO entities like Web Service Modeling Toolkit (WSMT), WSMO Studio.

3.3 Healthcare Projects based on Semantic Web Services

There are many healthcare projects that are using semantic web services. Some of the projects are explained below to shed light on the importance of semantic web services in the web service discovery and composition for bringing automation in the system.

COCOON is a web service technology based project aimed at reducing medical errors [32]. This project works on resolving the problem of integration in healthcare domain. The problem of integrating components from service discovery to service composition is handled. It is a WSMO compliant project and uses WSMO compliant service discovery engine for resolving the service discovery issue. In COCOON the most appropriate services are discovered and used by the specialist hence providing better healthcare services.

Artemis is another project based on semantic web services for the semantic discovery and composition of services. It uses OWL-S as the approach for implementing semantic web services. HL7 is used as a standard for communication of messages for sender to the receiver. Artemis used OWL mapping tool (OWLmt) for the communication between sender and receiver providing semantic interoperability. OWLmt works as a mediator between sender and receiver by comparing sender ontology instances and receiver ontology instances with each other for making possible the communication [32].

Plug and Play Electronic Patient Records (PPEPR) is a semantic SOA based platform that has the objective of integrating the heterogeneous Electronic Patient Records (EPR's). The main objective of PPEPR platform is to resolve heterogeneities related to data, process and service level [30]. The integration in PPEPR project is based on SOA, web services and semantics. It is based on different ontologies that include functional ontology, message ontology and mapping ontologies. Also it uses adapter framework for the conversion of message into goal form which is then provided to the semantically enabled middleware. The initial prototype of PPEPR tackles the heterogeneities between two types of HL7 standard HL7 V2 and HL7 V3. The architecture of PPEPR project is shown in Figure 3.3.

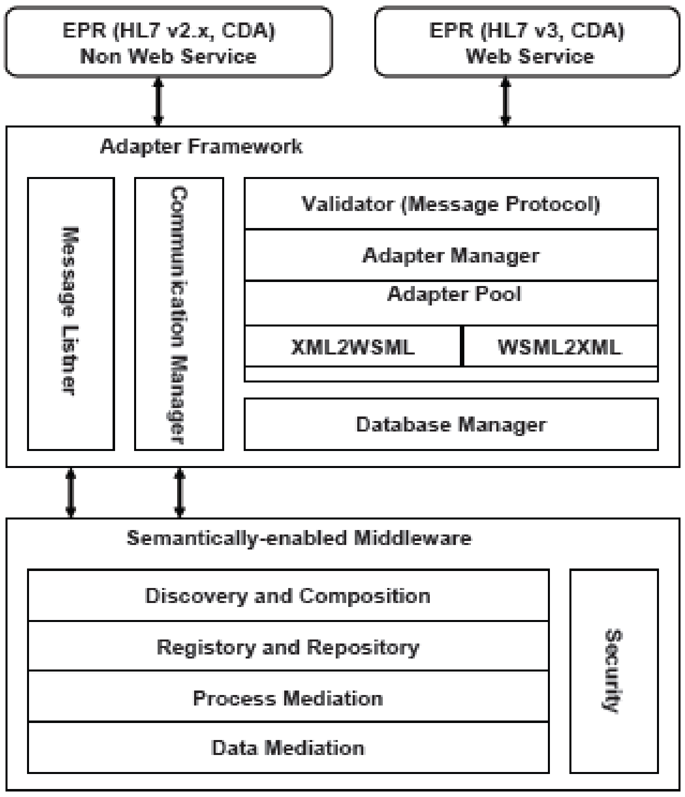

Figure 3.3: PPEPR Architecture
[30]

3.4 Critical Analysis

WSMO is considered more appropriate compared to OWL-S because it handles heterogeneity problem in a more comprehensive way. OWL-S doesn't support all the aspects related to web services while WSMO supports almost all aspects of web services. OWL-S provides no explicit distinction between choreography and orchestration and thus has no internal mechanism to manage workflows of different processes. Therefore OWL-S is always dependent on external work for defining workflow of process. Due to this OWL-S only supports one way to interact with service as Service Model which is defined per service. WSMO provides multiple interfaces to interact with service therefore handling choreography and orchestration effectively. Service provider and service requestor are not separated in OWL-S thus it lacks higher level of the degree of integration of functionalities. Profile ontology is used to support both the service provider and service requestor. On the other hand, in WSMO, the service requestor is handled through WSMO Goal while the service provider is handled by WSMO web service. WSMO also handles data and process heterogeneities with the help of Mediator, a top level entity of WSMO handling heterogeneities. OWL-S has no concept like mediator for handling heterogeneity [16]. COCOON project has deficiency as it doesnot support the transformation of International Classification of Diseases (ICD) terminologies transformation into ontologies for handling semantic data interoperability [32]. Also there is no mentioning of handling semantic process interoperability which is very important with semantic data interoperability. PPEPR project addresses data and process mediation in HL7 domain but is only related to the conversion between HL7 V2 and HL7 V3 messages. It is only based on the adapter that helps in the conversion of messages from HL7 V2 to HL7 V3 and vice versa.

It doesn't discuss much about the flow of interactions in HL7 V3 and how it will work in achieving certain goals. In contrast, the proposed system brings true semantic interoperability to HLH system. It uses the HL7 message generation and parsing components of HLH for incorporating semantics in the system.

Chapter 4

HLH WSMO Integration

In order to achieve the objective of semantic process interoperability, HLH Studio and WSMO architectures should be integrated. Semantic process interoperability is achieved by combining the parsing, discovery, selection, invocation, reasoning and grounding functionalities from WSMF (a Semantic SOA framework) to counter part HL7 artifacts.

4.1 HLH Studio Architecture

HLH Studio was developed for end to end message communication. Its architecture is based on the generation, storing, parsing, transportation and mapping of HL7 messages. The existing architecture of HLH Studio has the following components as shown in Figure 4.1.

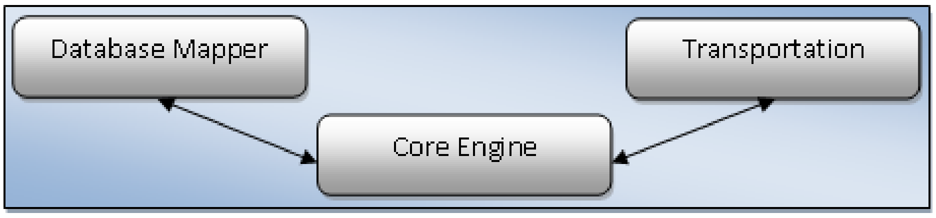

Figure 4.1: Existing HLH Studio Architecture

HLH Core Engine is responsible for the HL7 message generation and parsing. Java SIG API is used for the generation and parsing of HL7 message. These two components are used in the proposed system for the automation of the whole process. **HLH Transportation** component is mainly responsible for HL7 message transportation purpose. Different protocols can be used for the transportation of message such as Minimal Lower Layer Protocol (MLLP) and SOAP/ ebXML over HTTP. **Database Mapper** component provides the mappings between HL7 RIM model with the database schema. The corresponding mapping code is generated for the specified HL7 message.

4.2 WSMO Goal Discovery & Selection

Seamless communication is the objective of the proposed system. This objective is achieved with the help of WSMO entity Goals. The client only needs to provide the goal and doesn't bother about the rest of the communication. The internal processing of this goal is shown in the Figure 4.2. Client has to provide only the desire which is then converted to WSML form to find out the Goal. The Goal is discovered from the Goal repository having an exact or major match. The semantic web services are discovered based on this goal. The discovered semantic web services are ranked selected for further processing. The information about the process is provided by the semantic web service composed of choreography and orchestration which helps to control the flow of information between services. The service is invoked and the message transfer takes place. The message is sent to the receiver by hiding all the details of discovery process thus achieving seamless communication.

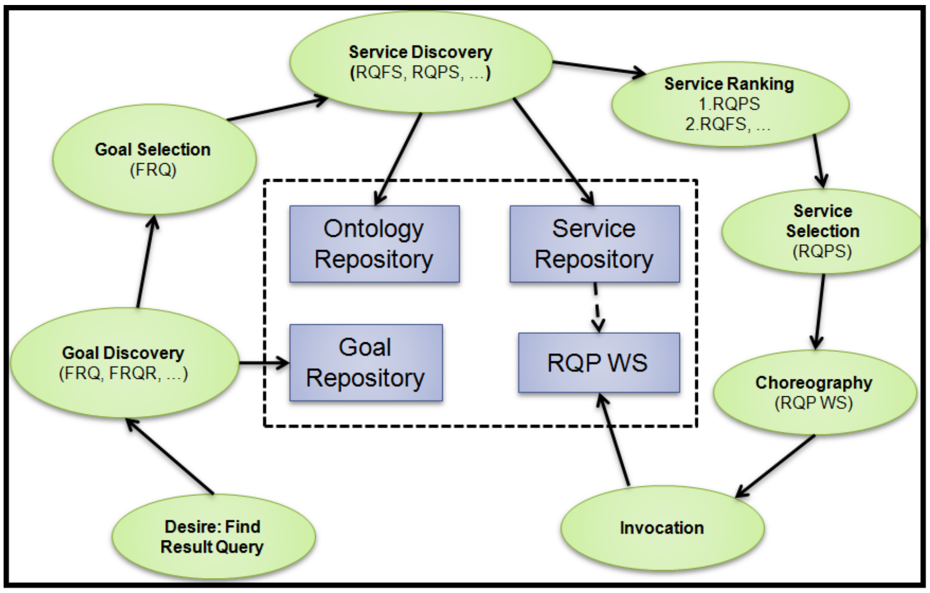

Figure 4.2: Seamless Communication with Goals

4.3 Proposed System Architecture

The proposed system is based on the integration of HLH and WSMO architectures. HLH Studio architecture is responsible for the HL7 message generation and parsing while the WSMO architecture is responsible for the overall process automation of the system with the help of semantic web services. The MessageGenerator and MessageParser components are used from the HLH. WSMO architecture is composed of Discovery, Selection, Ranking, Parsing, Composition, Reasoning, Grounding and Invocation components.

The architecture specified in Figure 4.3 is the abstract model of the system. The description of the components in the system are as follows;

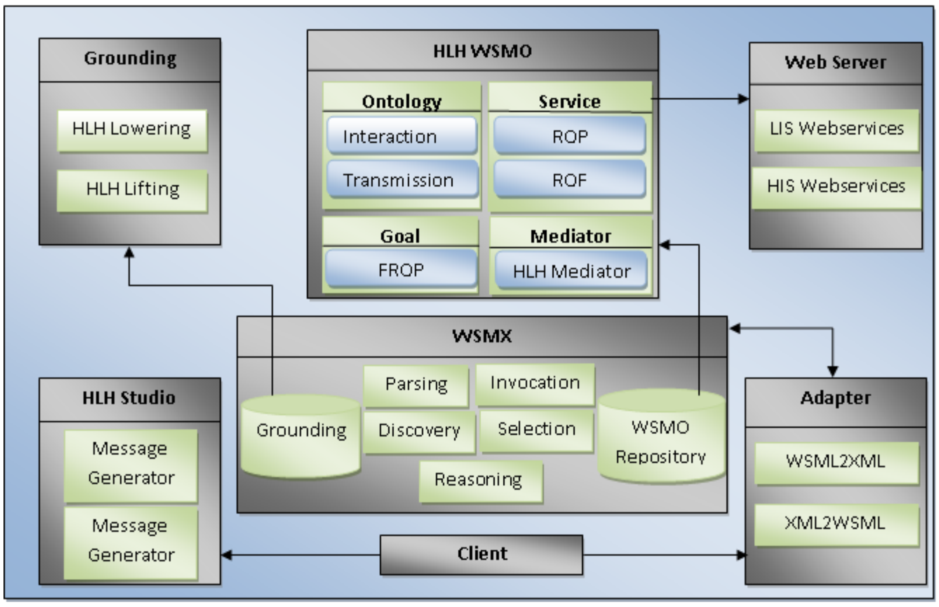

Figure 4.3: Proposed System Architecture

Message Generator

The MessageGenerator component is used to generate the HL7 V3 message using JavaSIG API. The generated message is then used by WSMX server for further processing HL7. RMIM is required for a particular message to be generated in XML form using JavaSIG API. The RMIM used for query message is shown in Figure 4.4. This RMIM is converted to HL7 message with the help of JavaSIG API. The pseudo code for the generation of message payload is shown in Figure 4.5.

A sample Result Query Placer payload in XML generated using this approach is shown in Figure 4.6.

Message Parser

The MessageParser component is used for checking the validity of HL7 message using JavaSIG API.

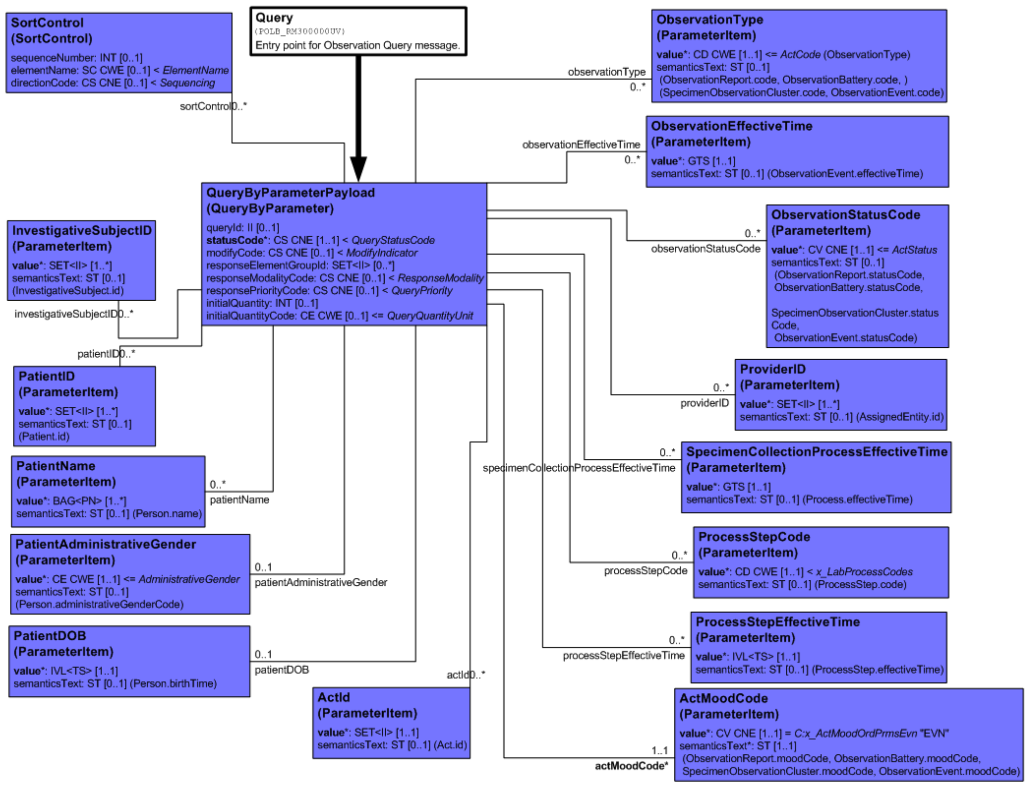
SortControl
(SortControl)
sequenceNumber: INT [0..1]
elementName: SC CWE [0..1] < ElementName
directionCode: CS CNE [0..1] < Sequencing
sortControl0..*
Query
(POLB_RM300000UV)
Entry point for Observation Query message.
QueryByParameterPayload
(QueryByParameter)
queryId: II [0..1]
statusCode*: CS CNE [1..1] < QueryStatusCode
modifyCode: CS CNE [0..1] < ModifyIndicator
responseElementGroupId: SET<II> [0..*]
responseModalityCode: CS CNE [0..1] < ResponseModality
responsePriorityCode: CS CNE [0..1] < QueryPriority
initialQuantity: INT [0..1]
initialQuantityCode: CE CWE [0..1] <= QueryQuantityUnit
InvestigativeSubjectID
(ParameterItem)
value*: SET<II> [1..*]
semanticsText: ST [0..1]
(InvestigativeSubject.id)
investigativeSubjectID0..*
patientID0..*
PatientID
(ParameterItem)
value*: SET<II> [1..*]
semanticsText: ST [0..1]
(Patient.id)
PatientName
(ParameterItem)
value*: BAG<PN> [1..*]
semanticsText: ST [0..1] (Person.name)
0..*
patientName
PatientAdministrativeGender
(ParameterItem)
value*: CE CWE [1..1] <= AdministrativeGender
semanticsText: ST [0..1]
(Person.administrativeGenderCode)
0..1
patientAdministrativeGender
PatientDOB
(ParameterItem)
value*: IVL<TS> [1..1]
semanticsText: ST [0..1] (Person.birthTime)
0..1
patientDOB
actId0..*
ActId
(ParameterItem)
value*: SET<II> [1..1]
semanticsText: ST [0..1] (Act.id)
observationType
0..*
ObservationType
(ParameterItem)
value*: CD CWE [1..1] <= ActCode (ObservationType)
semanticsText: ST [0..1]
(ObservationReport.code, ObservationBattery.code,)
(SpecimenObservationCluster.code, ObservationEvent.code)
observationEffectiveTime
0..*
ObservationEffectiveTime
(ParameterItem)
value*: GTS [1..1]
semanticsText: ST [0..1] (ObservationEvent.effectiveTime)
0..*
observationStatusCode
ObservationStatusCode
(ParameterItem)
value*: CV CNE [1..1] <= ActStatus
semanticsText: ST [0..1]
(ObservationReport.statusCode,
ObservationBattery.statusCode,
SpecimenObservationCluster.status
Code,
ObservationEvent.statusCode)
0..*
providerID
ProviderID
(ParameterItem)
value*: SET<II> [1..*]
semanticsText: ST [0..1] (AssignedEntity.id)
0..*
specimenCollectionProcessEffectiveTime
SpecimenCollectionProcessEffectiveTime
(ParameterItem)
value*: GTS [1..1]
semanticsText: ST [0..1] (Process.effectiveTime)
0..*
processStepCode
ProcessStepCode
(ParameterItem)
value*: CD CWE [1..1] < x_LabProcessCodes
semanticsText: ST [0..1] (ProcessStep.code)
0..*
processStepEffectiveTime
ProcessStepEffectiveTime
(ParameterItem)
value*: IVL<TS> [1..1]
semanticsText: ST [0..1] (ProcessStep.effectiveTime)
1..1
actMoodCode*
ActMoodCode
(ParameterItem)
value*: CV CNE [1..1] = C:x_ActMoodOrdPrmsEvn "EVN"
semanticsText*: ST [1..1]
(ObservationReport.moodCode, ObservationBattery.moodCode,
SpecimenObservationCluster.moodCode, ObservationEvent.moodCode)

Figure 4.4: Result Query RMIM in HL7

Adapter

The adapter component is used for the transformation of HL7 XML message in to WSML form. This WSML is then considered as a goal and is provided to WSMX for the discovery and invocation of web services. The desire of the client is first converted to WSML form for the seamless communication to take place and then on the way back the conversion from WSML to XML take place.

```
public Object generateMessagePayload()
{
      // code for QuerybyParameterPayload
      QueryByParameter queryByParameterPayload = new QueryByParameterImpl();
      II qID = IIimpl.valueOf("192","","");
      queryByParameterPayload.setQueryId(qID);
      queryByParameterPayload.setStatusCode(CSimpl.valueOf("active",
      "getCodeSystem"));

      //clone for Query by Parameter Payload
      queryByParameterPayload.setCloneCode(CSimpl.valueOf("queryByParameterPayload",
      "getCodeSystem"));

      // Code for PatientID
       ParameterItem patientID = new ParameterItemImpl();
        II pID = IIimpl.valueOf("2","","");
        patientID.setValue(pID);

      //add clone of patientID
       patientID.setCloneCode(CSimpl.valueOf("patientID", "getCodeSystem"));

      //adding queryByParameter with patientID
       queryByParameterPayload.addParameter(patientID);

      // Code for patient name
       ParameterItem patientName = new ParameterItemImpl();

       BAG<EN> name = null;
       name = DatatypeTool.EntityNameTool.setPrefixName(name, "Mr");
       name = DatatypeTool.EntityNameTool.setGivenName(name, "Robert");
       name = DatatypeTool.EntityNameTool.setFamilyName(name, "Zimmerman");
       name = DatatypeTool.EntityNameTool.setSuffixName(name, "VII");
       patientName.setValue(name);

      //adding queryByParameter with patientName
      queryByParameterPayload.addParameter(patientName);

      //add clone of Patient Name
      patientName.setCloneCode(CSimpl.valueOf("patientName", "getCodeSystem"));

      // Code for act mood code
      ParameterItem actMoodCode = new ParameterItemImpl();
      actMoodCode.setValue(CSimpl.valueOf("EVN", "1.22.333.4446"));

      //add clone of actMoodCode
      actMoodCode.setCloneCode(CSimpl.valueOf("actMoodCode", "getCodeSystem"));

      //adding query by parameter with act mood code
      queryByParameterPayload.addParameter(actMoodCode);

       return queryByParameterPayload;
}
```

Figure 4.5: HL7 RQP Message Payload Code

Parser

This component validates the WSML description files and that description can be then stored persistently in the Resource Man-

```
<?xml version="1.0" encoding="UTF-8"?> <QueryByParameterPayload
xmlns="urn:hl7-org:v3"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <queryId root="192" extension="" assigningAuthorityName="" displayable="false"/>
   <statusCode code="active"/>
   <actMoodCode>
      <value code="EVN"/>
   </actMoodCode>
   <patientID>
      <value xsi:type="II" root="2" extension="" assigningAuthorityName="" displayable="false"/>
   </patientID>
   <patientName>
      <value xsi:type="EN">
         <prefix>Mr</prefix>
         <given>Robert</given>
         <family>Zimmerman</family>
         <suffix>VII</suffix>
      </value>
   </patientName>
</QueryByParameterPayload>
```

Figure 4.6: HL7 RQP Payload

ager. Message Parser component is related to parsing of HL7 message in XML form while the parser component in WSMX is related to the parsing of WSML files.

Discovery

This component is responsible for the discovery of web services on the basis of goal provided. It matches the capability of the goal with the web services and rank different services for selection.

Selection

This component selects the web service for further action to be taken by the invocation component. This component selects the web service after the ranking component ranks web services on the basis of matching.

Invocation

This component is responsible for actual invocation of the web service for the transfer of the messages. The choreography of the semantic web service contains state signatures which has grounding for the end point web service to be invoked.

HLH WSMO Repository

The WSMO entities related to HL7 are stored in the HLH WSMO Repository. Ontologies include the Interaction and Transmission Ontology, web services contains the Result Query Placer and Result Query Filler web service and goals contains one related Find Result Query Placer.

Grounding

This component is responsible for the conversion from XML to WSML and vice versa. There are two components that are responsible for the conversion: Lifting and Lowering. Both lowering and lifting is performed using XSLT transformation. Lowering is done by converting SOAP request into RDF and then to WSML for further processing by WSMX. Lifting is done the opposite to Lowering and in lifting the WSML is converted to RDF and then to SOAP Response. Grounding component is composed of HLH Lifting and HLH Lowering subcomponents used for transformation.

Chapter 5

System Implementation

We used Web Service Modeling Toolkit (WSMT) for modeling WSMO entities and SOAP UI [1]tool for testing end-point web services and entrypoints. The main entities implemented for HLH Repository component and WSMO grounding component are explained below:

5.1 Interaction Ontology

Interaction ontology contains all the process artifacts and describes how they are associated with each other. Interaction ontology helps in automation of the process by finding out from the message contents suitable application roles for participation in communication. It helps in assigning responsibilities to these application roles like message sending to one application role and receiving (replying if necessary) to other application role. Also the interactions that will take place between these application roles are identified by the interaction ontology. It helps in identifying the message type associated with the particular application roles and interactions for transferring in the message. The ontology contains classes, subclasses, and properties of those classes, restrictions and instances of the classes.

[1]http://www.soapui.org/

5.1.1 Classes

There are four levels of classes that are included in interaction ontology. The main classes include the Application Role, Interaction, Message Type and Trigger Event. These main classes are shown in Figure 5.1.

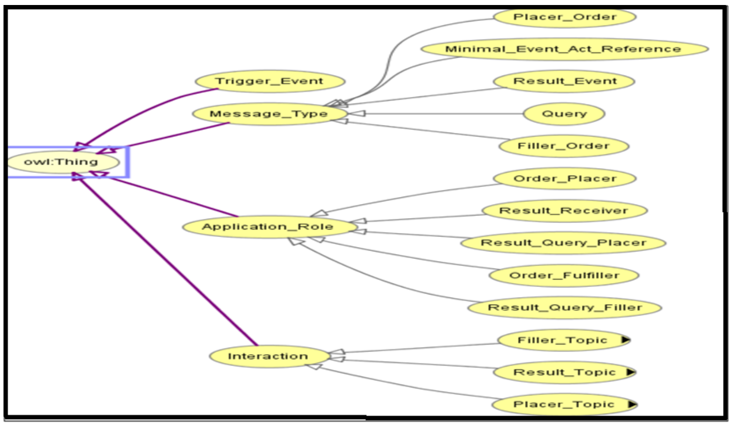

Figure 5.1: Classes in Interaction Ontology

5.1.2 Properties

Properties are defined for the classes to build associations between them. We can also restrict a relation by specifying its Domain and Range [1]. Properties short description is provided in Table 5.1;

Table 5.1: Interaction Ontology Properties

Property	**Domain**	**Range**	**Example**
playsRoleIn	Application Role	Interaction	Order Placer *playsRoleIn* Order Fulfillment Request
initiatedBy	Interaction	Trigger Event	Order Fulfillment Request is *initiatedBy* a Trigger Event
senderRole	Interaction	Sender Application Role	Order Fulfillment Request has *senderRole* Order Placer
receiverRole	Interaction	Receiver Application Role	Order Fulfillment Request has *receiverRole* Order Filler
transferredBy	Message Type	Interaction	Placer Order is *transferredBy* Order Fulfillment Request
inverseOf	Interaction	Interaction	Find Result Query has *inverseOf* Find Result Query Response

5.1.3 Property Restrictions

Scope of certain properties is restricted at the lower class hierarchy level. Table 5.2 shows the restrictions on the process artifacts. These restriction show the relationship of each of the process artifacts with others and how they work.

5.1.4 Hierarchy of Classes

There are three levels of hierarchy in interaction ontology. The first level contains core classes such as Application Role, Interaction, Message Type and Trigger Events. The application roles in HL7 V3 laboratory domain are then further categorized in to different types like: Order Placer, Order Fulfiller, Result Query Placer, Result Query Filler and Result Receiver. The interactions are also further divided into three sub-categories: Order, Promise and Result. These sub-categories of interactions are then divided in to further sub categories related to interactions of Order, Promise or Result. Each interaction is initiated by a trigger event and its information is stored in the trigger event class. The Message Type class represents different message types. The interactions are also categorized by message types. Every interaction in the laboratory domain must follow one of the message types which are subcategories in the interaction ontology like: Minimal Event Act Reference, Query, Placer Order, Filler Order and Result Event. The interaction ontology is shown in Figure 5.2.

5.2 Transmission Ontology

Transmission Ontology stores the information related to the message to be transferred. Its information is based on the HL7 Message as shown in Figure 5.3. HL7 Message is divided into

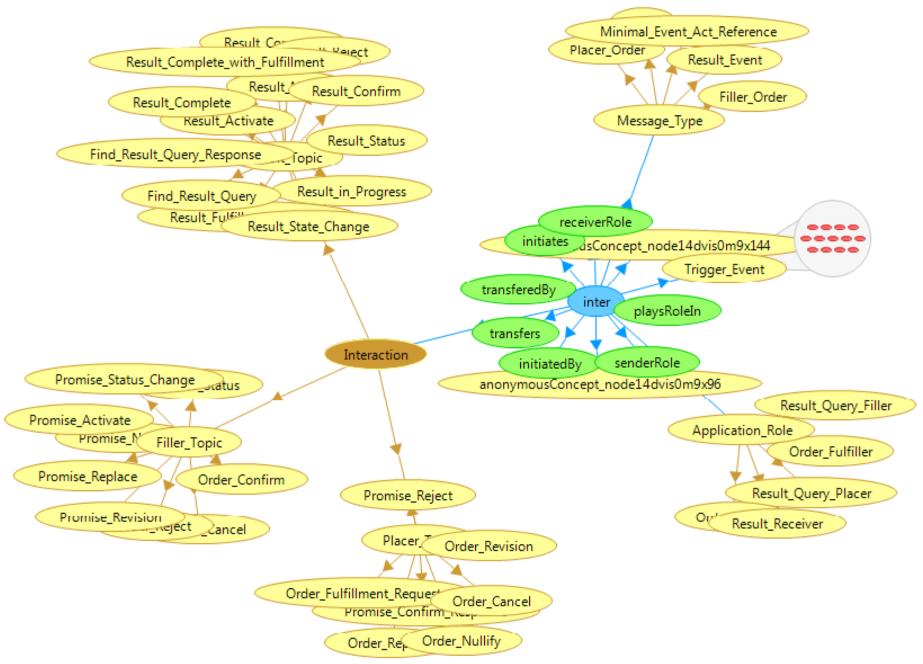

Figure 5.2: Interaction Ontology

three categories Message Payload, Control Act Wrapper and Transmission Wrapper. Transmission Ontology takes information related to the process artifacts from HL7 message and helps in the process. It helps in extracting the information related to the process artifact in the message which can then be used by interaction ontology.

5.3 Result Query Placer/ Result Query Filler Web Service

HLH Repository also contains information related to semantic web services. We initially created result query placer and result query filler semantic web services for our scenario. The application role concept in HL7 V3 is taken as semantic web services.

The application roles in HL7 V3 represent logically related functionality of the system and conformance of the system is

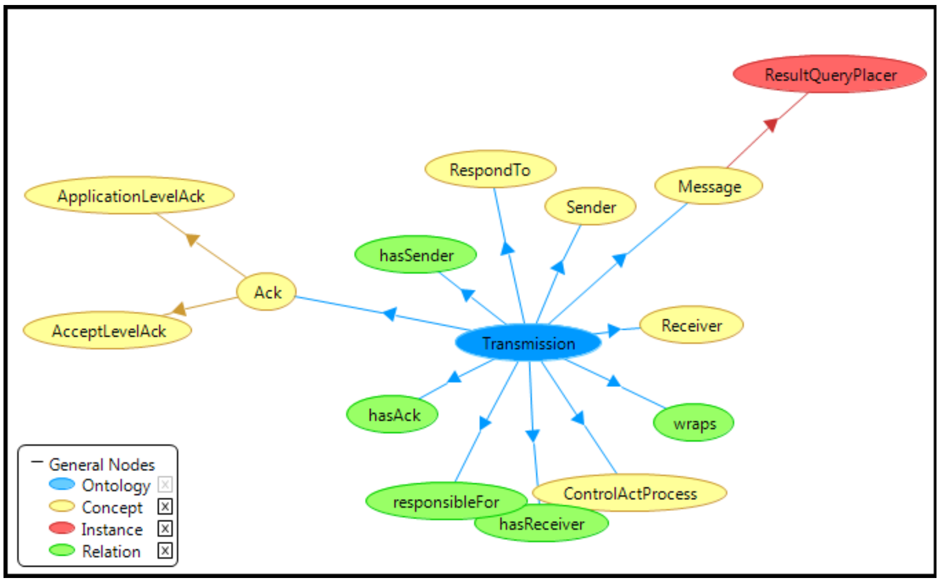

Figure 5.3: Transmission Ontology

based on implementation of various application roles. So keeping in view this definition and following HL7 web service basic profile, there is one to one correspondence of application roles and web services [33]. The proposed system implements application roles as semantic web services. Result Query Placer (RQP) is application role implemented as semantic web service in the proposed system which is used for querying about the status of the result at current point of time. Figure 5.4 shows the Result Query Placer web service in WSML form.

Result Query Filler is another application role as semantic web service that is used for giving the response related to the status of the result queried.

5.4 Find Result Query Goal

Find Result Query Goal is used for the discovery of Result Query Placer web service. The client has to provide information in the form of find result query goal that is then used for the discov-

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-rule"
namespace { _"http://www.hl7.seecs.edu.pk/services#",
    transmission _"http://seecs.hl7.edu.pk/transmission#",
    dc _"http://purl.org/dc/elements/1.1/",
    interaction _"http://hl7.seecs.edu.pk/inter#",
    discovery _"http://wiki.wsmx.org/index.php?title=DiscoveryOntology#",
    wsml _"http://www.wsmo.org/wsml/wsml-syntax#" }
webService ResultQueryPlacer importsOntology interaction#inter
capability ResultQueryPlacerCapability
    nonFunctionalProperties
      discovery#discoveryStrategy hasValue discovery#HeavyweightDiscovery
        dc#description hasValue "Result Query Placer interaction with
        message payload, transmission and control act wrapper"
    endNonFunctionalProperties
    precondition RQPPre
        definedBy
        ?message memberOf transmission#Message
        and ?message[transmission#interactionID hasValue "Find Result Query"]
        and transmission#ControlActProcess(?message, ?controlactprocess).

    postcondition RQPPost
        definedBy
        ?message memberOf transmission#ApplicationLevelAck
        and ?message[transmission#interactionID hasValue "Find Result Query Response"]
        and transmission#ControlActProcess(?message, ?controlactprocess).

interface RQPInterface
 choreography RQPChoreography
   stateSignature RQPstateSignature

   importsOntology interaction#inter

    in concept interaction#Find_Result_Query withGrounding _
    "http://localhost:8080/RQPSimpleTesting/RQPSimpleTestWSService?
    wsdl#wsdl.interfaceMessageReference(
    RQPSimpleTestWSPortType/RQPSimpleTestWSOperation/in0)"
```

Figure 5.4: ResultQueryPlacer Web Service

ery, ranking and selection of semantic web services which are then further used for the invocation of end point web services. Figure 5.5 shows the result of SOAP UI tool used to test achieve-Goal entry point.

5.5 HLH Lifting

In order to run a working scenario we carried out the transformation of SOAP Request form WSML to XML with XSLT. The processing will then take place for the data to be transferred by the web service.

5.6 HLH Lowering

In addition to lifting the lowering should also be done when the web service SOAP response takes place. The SOAP response is converted from XML to WSML by XSLT transform. The conversion takes place because WSMX only understands WSML and for the semantics to take effect the SOAP response is to be provided to the WSMX. After further processing the message is then communicated to the Adapter component on its way back to the client.

```
<se:Envelope xmlns:se="http://schemas.xmlsoap.org/soap/envelope/"
                       xmlns:ws="http://webservices.deri.ie">
  <se:Body>
    <web:achieveGoal>
      <ws:wsmlMessageGoal>http://hl7.seecs.edu.pk/goals#FRQGoal
                       </ws:wsmlMessageGoal>
    </ws:achieveGoal>
  </se:Body>
</se:Envelope>
```

Figure 5.5: SOAP Message for AchieveGoal Entrypoint

During system implementation we have faced some challenges regarding WSMX server. Building prototype of the WSMX 1.0 was quite complex in terms of missed jars and POMs files. Moreover, existing POMs were also updated to run according to the current environment. Due to lack of documentation, it was very difficult to understand and properly use the existing entry point for our scenario. For this purpose we have consulted WSMX technical team several times to resolve the issues.

Table 5.2: Restrictions on Classes and Properties

Class/Property	SubClass	Restriction
	Order Placer & Order Fulfiller	• ∀ playsRoleIn only (Result_Reject or Result_Status or Result_in_Progress or Result_Complete_with_Fulfillment or Result_Corrected or Result_Nullify or Result_Confirm) • ∀ playsRoleIn only (Order_Fulfillment_Request or Order_Revision or Order_Cancel or Order_Nullify or Order_Replace or Promise_Reject) • ∀ playsRoleIn only (Order_Reject or Order_Confirm or Promise_Activate or Promise_Revision or Promise_Cancel or Promise_Replace or Promise_Nullify or Promise_Nullify or Promise_Status_Change or Promise_Status)
Application Role/ playsRoleIn	Result Query Placer & Result Query Filler	• ∀ playsRoleIn only (Find_Result_Query or Find_Result_Query_Response)
	Order Fulfiller & Result Receiver	• ∀ playsRoleIn only (Result_Activate or Result_Complete or Result_State_Change)
	Result Status, Result Nullify, Result in Progress,Result Corrected, Result Confirm, Result Complete With Fulfillment	• ∃ receiverRole some Order_Placer • ∃ senderRole some Order_Fulfiller
	Result Reject	• ∃ receiverRole some Order_Fulfiller • ∃ senderRole some Order_Placer
	Result State Change, Result Activate, Result Complete	• ∃ receiverRole some Result_Receiver • ∃ senderRole some Order_Fulfiller
Interaction/senderRole & receiverRole	Find Result Query Response	• ∃ receiverRole some Result_Query_Placer • ∃ senderRole some Result_Query_Filler
	Find Result Query	• ∃ receiverRole some Result_Query_Filler • ∃ senderRole some Result_Query_Placer
	Placer Topic (all subclasses)	• ∀ receiverRole only Order_Fulfiller • ∀ senderRole only Order_Placer
	Filler Topic (all subclasses)	• ∀ receiverRole only Order_Placer • ∀ senderRole only Order_Fulfiller
	Minimal Event Act Reference	• ∀ transferedBy only (Result_Confirm or Result_Reject)
	Query	• ∀ transferedBy only Find_Result_Query
Message Type/ transferedBy	Result Event	• ∀ transferedBy only (Result_Status or Result_State_Change or Result_Nullify or Result_in_Progress or Result_Corrected or Result_Complete_with_Fulfillment or Result_Complete or Result_Activate or Find_Result_Query_Response)
	Placer Order	• ∀ transferedBy only (Order_Fulfillment_Request or Order_Revision or Order_Cancel or Order_Nullify or Order_Replace or Order_Reject or Order_Confirm)
	Filler Order	• ∀ transferedBy only (Promise_Reject or Promise_Confirm_Response or Promise_Activate or Promise_Revision or Promise_Cancel or Promise_Replace or Promise_Nullify or Promise_Status_Change or Promise_Status or Promise_Status)

Chapter 6

System Evaluation

The proposed system is part of the HLH project, therefore it is evaluated with the current system that is developed and tested at CITI Lab, one of the renowned laboratories at Pakistan. The current system is deployed with semantics while our proposed system tries to remove its deficiencies by handling ti through semantics.

6.1 Comparison with baseline Framework

The proposed systems have some advantages over the currently HLH system. Current solution is based on MLLP protocol for the communication of messages while the proposed system is based on Web Services that provides scalability, flexibility and cost effectiveness to the system. Manually most of the things are handled in the current system while the proposed system provides automation and seamless communication. This results in the timely delivery of medical information transfer to the receivers as compare to the current system. HL7 V3 claims semantic interoperability but provides only data interoperability, process interoperability is still a grey area. Current system only focuses on limited semantics related to data using terminologies while process is not mentioned. The proposed system has the advantage over current system related to the interoperability as it focuses on process interoperability as well and has handles the behavioral aspects of HL7 V3 as well. As the current system is not based on web services and uses MLLP protocol therefore integration of other domains with the current system is very hard. Due to the nature of web services and its architecture in the form of SOA and the advantage of network and platform independence, the integration in the proposed system is very easy as compare to current system.

Table 6.1: Comparison of HLH and the Proposed System

Parameters	Current HLH System	Proposed System
User Role	Intervene in most of the aspects for message communication • Delay in transfer of information	Only provides desire in the form of goal • Timely delivery of information
Automation Level	Manual • Client is responsible for performing most of the functionalities	Semi Automatic • Client only needs to provide goal and process will take place automatically
Semantics	Lacks Semantics (minimal data related semantics) • Interoperability related to data only catered	Data and process related semantics • Interoperability related to both data and process are catered
Communication Protocol	MLLP • Hard integration	Web Services • Flexible • Cost Effective • Easy Integration
Workflow	Manual	Automated

6.2 Evaluation Criteria

The ultimate success criterion for any business process management system is automation of the workflow. Our proposed system implements two different types of workflows based on their requirement of human intervention (similar to BPEL4People). HL7 Test Order interactions, for example, by design require human input; Result Query, on the contrary, doesn't require human intervention and is therefore completely automated. The heterogeneities in workflows across multiple healthcare organizations are resolved automatically by our developed system by taking into account that data loss does not take place.

As a quantitative evaluation criterion we considered In-Patient Encounter scenario (a single patient visiting multiple health providing unit/labs). The metric to be observed and evaluated in this scenario was 100% automation of the workflow (except where human intervention is required by design). The scenario is explained as under:

A physician orders to admit a patient for particular treatment (such as liver resection) in a hospital. The patient encounter is scheduled on particular date. Before getting into surgery, the patient goes through multiple diagnostic-tests (related to radiology and pathology) at the same hospital and two other diagnostic laboratories. After getting the desired observation values, the patient surgery is performed.

6.3 Evaluation Procedure

The main objective of our proposed approach is automation in the workflow. Our focus, therefore, remains automatic service discovery and service invocation. The services involved in the workflow are Patient Admission, Patient Admission Scheduler, Lab Test Order, Lab Test Result and Patient Discharge Services.

The lab test order and lab test result services are related to HL7 laboratory domain while patient related services are related to HL7 patient administration domain. The information of patient administration and laboratory domain services were incorporated in the Interaction Ontology. It can also be extended in future for other HL7 domains thus flexibility is achieved.

6.4 Experiment Design and Execution

The flow starts from the discovery of Patient Admission Service. Once the Patient Admission Service is invoked, it dynamically finds appropriate Patient Scheduling service to schedule patient admission. After getting into Patient Admission Scheduler service, all the related information required for particular encounters are passed to the particular hospital. It may include scheduling the tests before going into proper encounter. After confirming the admission, the Admission Service of the assigned hospital is accessed to verify the schedule encounter and provide the patient with all required facilities. In this case, the laboratory test order service will dynamically invoke to schedule tests for the patient. After getting observation from the laboratory (a local diagnostic laboratory, CITI lab, in our case), the results are returned back to encounter service for further processing of encounter. After encounter is finished, the discharge service is invoked to post the discharge summary of the patient.

6.5 Evaluation Metrics

The current system can be evaluated on the basis of flexibility, interoperability and timely delivery of medical information.

Flexibility

Flexibility is the ability to measure the ease with which a system adds other components different than those for which it was specifically designed [3]. Interaction Ontology provides flexibility to the proposed system by easily integrating process artifacts of other domains in addition to laboratory domain. No compilation of the system is required because the information related to process artifacts is added to the ontology. In addition to flexibility provided by ontology, the proposed system flexibility is further strengthened because of its compliance to SOA. Flexibility is the requirement of the proposed system because HL7 consists of various domains necessary to be incorporated for better patient care.

Timely Delivery

Due to less human intervention, the proposed system brings automation in healthcare information exchange thus provides seamless communication. This automation leads to timely delivery of information from sender to the receiver for further processing.

Interoperability (Conformance to HL7)

The system is based on semantic framework but yet the message is conformed to HL7 specifications. Initially the message is in the XML form, it is converted to WSML form for further processing by the system. After processing by the system the message is communicated to the receiver in the same form as it was initially therefore no change in the HL7 message has taken place.

Chapter 7

Conclusion and Future Work

This chapter concludes the overall description related to the system from its analysis to implementation. HL7 V3 provides semantic interoperability but it is limited to data interoperability. This thesis extends this ability with semantic process interoperability. Along with semantic data interoperability it will contribute towards true semantic interoperability.

7.1 The Proposed System Benefits

The proposed system plays a key role to realize healthcare systems with HL7 V3 compliance in achieving true semantic interoperability. It provides the following benefits

Seamless Communication

The sender and the receiver are not concerned with the complexity of message transfer. The communication takes place seamlessly between the communicating devices. The client is only required to show his desire in the form of goal. The goal will then lead to the service discovery and service composition seamlessly.

Timely delivery of healthcare services

The proposed system provides timely delivery of healthcare services to the patients. Patient doesn't have to wait for long unlike traditional HL7 approach. The timely delivery is achieved because of the automation in the process which is the result of embedding semantics in the system. This timely delivery results in the saving of costs and probably the lives.

Patient Satisfaction

Patient is only satisfied when the healthcare services are provided to them with minimum delay. The proposed system provides abrupt healthcare services thus leading to patient satisfaction. Also the patient information can be delivered to specialist in less time therefore effective care is provided to the patient.

Cost Effectiveness

The proposed system is based on semantic SOA based on services. These services are reusable and also other services can be embedded easily in the SOA architecture therefore the system is cost effective.

Process Automation

The proposed system provides semantics to services which leads to automation. The whole process works on the automatic discovery of goals and services on the basis of desire shown by the client or user.

7.2 Conclusion

The proposed system is a cost effective, flexible and interoperable solution with strong emphasis on the importance of semantic process interoperability with semantic data interoperability. It focuses on the timely delivery of information for providing better health to patients. It emphasizes on the importance of semantic process interoperability with semantic data interoperability. Semantic interoperability is not complete without semantic process interoperability. It also shows that WSMF is the most suitable framework for providing semantic web services that leads to bringing automation in the system. The behavioral aspects of HL7 V3 are combined with the terminologies in order to effectively use the standard and provide interoperability which is the main concern of healthcare domains nowadays. The proposed systems is dependent on WSMX for its execution semantics but WSMX is a prototype version which leads to the deficiency of this system.

7.3 Future Work

The proposed system will play an important role in bringing semantics in the Electronic Health Records (EHR) systems. Also mediators in WSMO architecture can be used for providing database mapping with the HL7 RIM model. The mediator component can also be used for HL7 V2 to V3 conversion which is the demand of most of the healthcare organizations nowadays. As most of the hospitals are based on HL7 V2 and cannot afford the paradigm shift due to extra cost therefore they require a system that can convert messages from HL7 V2 to V3 and vice versa for interoperability purpose. Mediator can provide the way in this use case and make the system interoperable.

Bibliography

[1] Owl web ontology language guide, w3c recommendation 10 february 2004. Website. `http://www.w3.org/TR/2004/REC-owl-guide-20040210/`.

[2] Semantic web enabled web services (swws). Website. `http://www.swsi.org/resources/swws-wsmf.pdf`.

[3] Standard glossary of software engineering terminology. Technical report, IEEE, September 1990.

[4] About hl7. Website, 2010. `http://www.hl7.org/about/index.cfm`.

[5] Hl7 v3 lab, r1, hl7 version 3 standard: Laboratory, release 1 last ballot. Website, 2010. `http://www.hl7.org/v3ballot2008jan/html/welcome/environment/index.htm`.

[6] Web services. Website, 2010. `http://www.globus.org/toolkit/docs/4.0/common/key/`.

[7] Muhammad Afzal, Maqbool Hussain, Hafiz Farooq Ahmad, and Arshad Ali. Design and implementation of open source hl7 version 3 for e-health services. In *Proceedings of 9th International HL7 Interoperability Conference (IHIC)*, pages 174–179, Crete Greece, October 2008.

[8] Steve Battle and et al. Semantic web services framework (swsf) overview. Technical report, W3C Member Submis-

sion, September 2005. http://www.w3.org/Submission/SWSF/.

[9] George Beeler and et al. Hl7 reference information model. ANSI/HL7 V3 RIM, R1-2003, HL7 Normative 2009.

[10] George W Beeler, Stan Huff, Wesley Rishel, Abdul-Malik Shakir, Mead Walker, Charlie Mead, and Gunther Schadow. Message development framework. Technical report, Version 3.3, Copyright 1999 by Health Level Seven, Inc., December 1999.

[11] Christoph Bussler and et al. Web service execution environment (wsmx). Technical report, W3C Member Submission, June 2005. http://www.w3.org/Submission/WSMX/.

[12] Liliana Cabral, John Domingue, Enrico Motta1, Terry Payne, and Farshad Hakimpour. Approaches to semantic web services: An overview and comparisons. In *In proceedings of the First European Semantic Web Symposium, ESWS*, pages 225–239, Heraklion, Crete, Greece, 2004. Springer.

[13] Emilia Cimpian. The web service execution environment (wsmx). Website. http://www.ocg.at/ak/ebusiness/files/WSMX.pdf.

[14] Jos de Bruijn and et al. Web service modeling language (wsml). Technical report, W3C Member Submission, June 2005. http://www.w3.org/Submission/WSML/.

[15] Jos de Bruijn and et al. Web service modeling ontology (wsmo). Technical report, W3C Member Submission, June 2005. http://www.w3.org/Submission/WSMO/.

[16] Jos de Bruijn, D. Fensel, M. Kifer, J. Kopeck, R. Lara, Holger Lausen, A. Polleres, D. Roman, J. Scicluna, and

I. Toma. Relationship of wsmo to other relevant technologies. Technical report, W3C Member Submission, June 2005.

[17] Alexander Dobrev and et al. The socio-economic impact of interoperable electronic health record (ehr) and eprescribing systems in europe and beyond. Technical report, European Commission, Information Society and Media, October 2009. `http://www.ehr-impact.eu/downloads/documents/EHRI_final_report_2009.pdf%`.

[18] D. Fensel and C.Bussler. The web service modeling framework wsmf. In *Electronic Commerce Research and Applications, Vol. 1, Issue 2, Elsevier Science B.V.*, 2002.

[19] W3C Semantic Annotations for Web Service Description Language Working Group. Semantic annotations for wsdl and xml schema. Technical report, W3C Recommendation, August 2007. `http://www.w3.org/TR/sawsdl/`.

[20] Karthik Gomadam, Ajith Ranabahu, and Amit Sheth. Sa-rest: Semantic annotation of web resources. Technical report, W3C Member Submission, April 2010. `http://www.w3.org/Submission/SA-REST/`.

[21] Farshad Hakimpour, Denilson Sell, Liliana Cabral, John Domingue, and Enrico Motta. Semantic web service composition in irs-iii: The structured approach. In *In CEC '05: Proceedings of the Seventh IEEE International Conference on E-Commerce Technology (CEC'05)*, pages 484–487, Washington, DC, USA, 2004. IEEE.

[22] Mick Kerrigan. Web service modeling toolkit (wsmt). Technical report, WSMX Working Draft, April 2005. `http://www.wsmo.org/TR/d9/d9.1/v0.2/20050425/`.

[23] Mick Kerrigan, Adrian Mocan, Martin Tanler, and Werner Bliem. Creating semantic web services with the web service modeling toolkit (wsmt). In *Proceedings of Workshop on Making Semantics Work for Business (MSWFB) at the 1st European Semantic Technology Conference (ESTC)*, Vienna, Austria, May 2007.

[24] Wajahat Ali Khan, Maqbool Hussain, Muhammad Afzal, Khalid Latif, Hafiz Farooq Ahmad, and A.M.Khattak. Towards semantic process interoperability. In *Proceedings of 10th International HL7 Interoeprability Conference (IHIC)*, pages 88–95, Kyoto, Japan, May 2009.

[25] Rubem Lara and et al. A conceptual comparison between wsmo and owl-s. Technical report, WSMO Working Draft, January 2005. `http://www.wsmo.org/2004/d4/d4.1/v0.1/20050106/d4.1v0.1_20050106.pdf`.

[26] David Martin and et al. Owl-s: Semantic markup for web services. Technical report, W3C Member Submission, November 2004. `http://www.w3.org/Submission/OWL-S/`.

[27] Blackford Middleton. New findings show that investment in standardized healthcare information exchange would deliver $87 billion in annual healthvare savings. Center for Information Technology Leadership, Febuary 2004. `http://www.citl.org/news/HIEI\_Findings.pdf`.

[28] Tom Miller. Siemens healthcare- workflow and solution division. `http://www.siemens.com/pool/de/investor_relations/finanzpublikationen/r%eden\_prasentationen/cmd2008/cmd\_feb\_2008\_miller.pdf`, February 2008.

[29] Lyndon Nixon and Elena Paslaru. State of the art of current semantic web services initiatives. Technical report, EU-IST Network of Excellence (NoE) IST-2004-507482 KWEB, June 2004. `http://knowledgeweb.semanticweb.org/semanticportal/deliverables/D2.4.ID%1.pdf`.

[30] Ratnesh Sahay, Waseem AKhtar, and Ronan Fox. Ppepr: Plug and play electronic patient records. In *Proceedings of the 2008 ACM Symposium on Applied Computing (SAC)*, Fortaleza, Ceara, Brazil, March 16-20 2008. ACM.

[31] Xiping Song, Beatrice Hwong, Gilberto Matos, Arnold Rudorfer, Christopher Nelson, Minmin Han, and Andrei Girenkov. Understanding requirements for computer-aided healthcare workflows: Experiences and challenges. In *International Conference on Software Engineering (ICSE)*, Shanghai, China, May 2006.

[32] Emanuele Della Valle, Dario Cerizza, Veli Bicer, Yildirak Kabak, Gokce Banu Laleci, and Holger Lausen. The need for semantic web service in the ehealth. In *In W3C workshop on Frameworks for Semantics in Web Services*, 2005.

[33] Transportation specification: Web service profile, release 2, hl7 v3 trans ws r2,committee ballot. `http://www.hl7.org/v3ballot2008jan/html/welcome/environment/index.htm`, January 2008.