BAŞKENT UNIVERSITY

INSTITUTE OF SCIENCE AND ENGINEERING

AN APPLICATION OF SERVICE ORIENTED ARCHITECTURE (SOA) APPROACH

DENİZ ÇOPUR

MASTER THESIS

2011

SERVİS ODAKLI MİMARİ YAKLAŞIMI UYGULAMASI

AN APPLICATION OF SERVICE ORIENTED ARCHITECTURE APPROACH

DENİZ ÇOPUR

Thesis Submitted

in Partial Fulfillment of the Requirements

for the Degree of Master of Science

in Department of Computer Engineering

at Başkent University

2011

This study named “An Application of Service Oriented Architecture (SOA) Approach” is accepted by our jury as **MASTER OF SCIENCE THESIS IN COMPUTER ENGINEERING** on 20/04/2011.

Chairman (Supervisor) :

(Prof. Dr. A. Ziya AKTAŞ)

Member :

(Asst. Prof. Dr. Mustafa SERT)

Member :

(Dr. Ali ARİFOĞLU)

...../05/2011

Prof. Dr. Emin AKATA

Director of Institute of

Science and Technology

ACKNOWLEDGEMENTS

The author would like to acknowledge the contributions of Prof. Dr. Ziya AKTAŞ (Supervisor), his father Tahsin Çopur and his mother Tuna Çopur for their invaluable guidance and perpetual patience. This thesis wouldn’t exist without their help.

ABSTRACT

AN APPLICATION OF SERVICE ORIENTED ARCHITECTURE APPROACH

Deniz Çopur

Başkent University Institute of Science and Engineering

Department of Computer Engineering

SOA (Service Oriented Architecture) is an architectural style of building applications based on composable, coarse-grained, loosely coupled autonomus software agents or components called services. A service provider system provides services containing one or more business operations, and service consumer systems communicate with the service provider by calling operations defined in the provided services. In this way, the service provider system and service consumer systems are integrated. The most important problem of the contemporary healthcare systems is integration. By integrating various healthcare systems, electronic health record belonging to an individual can be shared in a secure way. The most important organization working on solving healthcare systems' integration problems today is IHE (Integrating Healthcare Enterprise). IHE targets solving integration problems of healthcare systems by creating various healthcare integration profiles. The key objective of this thesis is to show that IHE profiles can be implemented using SOA.

KEY WORDS: integration, service oriented architecture, web services

Supervisor: Prof. Dr. A. Ziya AKTAŞ, Başkent University, Computer Engineering Department.

ÖZ

SERVİS ODAKLI MİMARİ YAKLAŞIMI UYGULAMASI

Deniz Çopur

Başkent Üniversitesi Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

SOA (Servis Odaklı Mimari) büyük ölçekli uygulamaların, servis adı verilen yazılım bileşenleri kullanılarak geliştirilmesidir. Servisler içlerinde bir ya da daha fazla sayıda süreç barındıran iri ölçekli, birbirlerine gevşek bağlı otonom yapılardır. Servis sunan uygulama, iş operasyonlarını bir ya da daha fazla sayıda servisler olarak sunar, ve servis tüketici uygulamalar, bu servisleri kullanarak servis sunan uygulama ile iletişim sağlar. Bu durumda, servis sunucu ve servis tüketici uygulamalar, servis odaklı mimari kullanarak entegre olmuş olur. Günümüzde sağlık alanı için geliştirilmiş uygulamaların en önemli sorunu entegrasyon sorunudur. Kişisel sağlık bilgisinin farklı sağlık sistemleri arasında gerektiğinde güvenli bir şekilde iletilmesi, bu sağlık sistemlerinin entegre olması ile sağlanabilir. Günümüzde sağlık sistemleri entegrasyonu alanında çalışan en önemli organizasyon IHE'dir (Integrating Healthcare Enterprise). IHE, çeşitli profiller oluşturarak, bu profillere uygun davranan sağlık sistemlerinin entegre olabilmelerini hedefler. IHE profilleri servis odaklı mimari kullanılarak gerçekleştirilebilir.

ANAHTAR SÖZCÜKLER: ağ servisleri,entegrasyon, servisodaklı mimari

Danışman: Prof. Dr. A. Ziya AKTAŞ, Başkent Üniversitesi, Bilgisayar Mühendisliği Bölümü.

TABLE OF CONTENTS

Page

ABSTRACT.....i

ÖZ.....ii

TABLE OF CONTENTS.....iii

LIST OF FIGURES.....vi

LIST OF TABLES.....vi

LIST OF ABBREVIATIONS.....vii

1 INTRODUCTION.....1

1.1 Definition of SOA.....1

1.2 SOA Related Terms.....3

1.2.1 Service.....3

1.2.2 Service provider.....3

1.2.3 Service consumer.....3

1.2.4 Contract.....3

1.2.5 End point.....4

1.2.6 Message.....4

1.3 A Brief History of SOA 4

1.4 Promises of SOA 1.....5

2 SERVICES.....10

2.1 Definition of a Service.....10

2.2 Characteristics of Services 10

2.2.1 Services share a formal contract.....11

2.2.2 Services are loosely coupled.....12

2.2.3 Services abstract underlying logic.....13

2.2.4 Services are composable.....14

2.2.5 Services are reusable.....15

2.2.6 Services are autonomous.....16

2.2.7 Services are stateless....17
2.2.8 Services are discoverable....18
3 WEB SERVICES....20
3.1 Definition of Web Services....20
3.2 A Typical Web Service Scenario....22
3.3 Characteristics of Web Services 24
4 ISSUES RELATED TO THE IMPLEMENTATION AND MANAGEMENT OF SOA....26
4.1 Approaches to Integration....26
4.1.1 Intelligent routing....26
4.1.2 Transformation....26
4.1.3 Tools for service monitoring....27
4.1.4 Advanced capabilities....27
4.1.5 State management....27
4.2 Approaches to Service Enablement....27
4.2.1 Service proxies....27
4.2.2 Service adapters....29
4.2.3 Unified information views....30
4.2.4 Data sources....31
4.2.5 Access and security....31
4.3 Technologies Relevant to SOA....33
4.3.1 ESB (Enterprise Service Bus)....34
4.3.2 BPM (Business Process Management)....38
4.3.2.1 Barriers to BPM....39
4.3.2.2 Properties of recent BPM software....41
5 AN APPLICATION....44
5.1 IHE (Integrating Healthcare Enterprise)....44
5.2 IHE Domains....46
5.3 IHE Profiles....47

5.4 Patient Demographics Query (PDQ) Profile.....49
5.4.1 PDQ roles.....50
5.4.2 PDQ sequence diagram.....51
5.4.3 PDQ process flow.....52
5.4.4 Transactions between actors.....55
5.4.4.1 Patient demographics query.....55
5.4.4.2 Patient demographics response.....56
5.5 Implementation Architecture.....57
5.6 Technologies Used In PDQ Implementation.....57
5.7 PDQ Service Implementation Details.....59
5.8 ESB Usage.....61
5.9 Client Implementation.....62
6 SUMMARY AND CONCLUSIONS.....63
6.1 Summary.....63
6.2 Conclusions.....64
6.3 Extensions of the Study.....66
REFERENCES.....67
APPENDICES.....71
APPENDIX **A.** HL7 MESSAGE SEGMENT DETAILS OF PATIENT DEMOGRAPHICS QUERY MESSAGE.....72
APPENDIX **B.** HL7 MESSAGE SEGMENT DETAILS OF PATIENT DEMOGRAPHICS RESPONSE MESSAG..... 78
APPENDIX **C.** PDQ QUERY AND RESPONSE EXAMPLES.....81
APPENDIX **D.** CD CONTAINING APPLICATION IMPLEMENTATION

LIST OF FIGURES

Figure 1. Web services scenario.....22

Figure 2. Enterprise Service Bus (ESB).....34

Figure 3. PDQ actors and transactions.....50

Figure 4. PDQ sequence diagram.....52

Figure 5. Some clinical use cases which require patient demographics Query.....53

Figure 6. Logical architecture of PDQ implementation.....58

LIST OF TABLES

Table 1. PDQ actors and transactions.....51

Table 2. PDQ Patient Demographics Request (HL7 QBP^Q22) segments.....56

Table 3. PDQ Patient Demographics Response (HL7 RSP^K22) segments.....57

Table A1. QDP segment.....73

Table A2. QPD-3 fields required to be supported.....74

Table A3. RCP segment.....76

Table A4. DSC segment.....77

Table B1. QAK segment.....79

Table B2. PID segment.....79

LIST OF ABBREVIATIONS

CORBA	Common Object Request Broker Architecture
DCOM	Distributed Component Object Model
DICOM	Digital Imaging and Communications in Medicine
EDI	Electronic Data Interchange
EHR	Electronic Health Record
ESB	Enterprise Service Bus
HL7	Health Level 7
IHE	Integrating the Health Enterprise
SOA	Service Oriented Architecture
BPEL	Business Process Execution Language
BPM	Business Process Management
PDQ	Patient Demographics Query
RMI	Remote Method Invocation
SOAP	Simple Object Access Protocol
UDDI	Universal Description, Discovery and Integration
WSDL	Web Services Definition Language
XML	Extensible Markup Language

1 INTRODUCTION

1.1 Definition of SOA

SOA (Service Oriented Architecture) is an architectural style of building applications based on composable, coarse-grained, loosely coupled autonomus software agents or components called services. Services are invoked by various service consumers, for that reason they should have well-defined interfaces. These interfaces should be based on standards, so that the service consumers and service providers may exist in different hardware platforms, operating systems, and developed using different programming languages, but still able to communicate.

SOA is concerned with the independent construction of business-aligned services that can be combined into meaningful, higher level business processes and solutions within the enterprise.

SOA definitions are very flexible and do not mandate use of specific communication protocols, transport mechanisms, message exchange semantics, programming languages etc. For services published by a service provider that can be invoked by a service consumer, these two systems must communicate. In SOA applications service providers and service consumers communicate using messages. But SOA approach does not mandate the format these messages should be in, what transport protocols should be used etc. For instance, service providers and service consumers can communicate using request/reply semantics or use asynchronous communication patterns, they can use HTTP, JMS (Java Messaging Service), SMTP (Simple Mail Transfer Protocol) or another transfer protocol, and they can exchange messages prepared in SOAP (Simple Object Access Protocol) format or some other message format as noted by Rotem [1].

In order to understand what SOA is, the concept of service should be well understood. Services are pillars on which the SOA systems are built. SOA is a modeling technique that groups related behaviors or functionality into software constructs or large components called services. The system acting as a service provider can publish one or more services, each having one or more behavior or functionality. A component of the system offering a service waits in a state of

readiness. Consumer components may invoke the service functionality by sending a message to the service according to the shared service contract [2].

The concept of SOA came about as a way of defining software architectures in a more sophisticated manner by paying greater attention to the exchanges amongst large software components. In addition, the method of service orientation places an emphasis on reusability by separating the interface of a function from its internal implementation. This form of separation makes service orientation an appropriate method for both heterogeneous and distributed architectures [3].

Popularity of SOA has increased during the last ten years. This popularity has come as a result of industry standards that identify interfaces and communication protocols associated with such services. These services are known as Web services and they have the capacity to greatly enhance interoperability which makes well designed services easier to integrate and reuse. When this is taken into account from an ICT perspective, it appears that the utilization of Web services tends to lower costs because it promotes reusability of services, and removes the need for proprietary software adapters. In addition, it can also speed time to market because it allows for parallel development of services [3].

Rotem[1] states that contemporary SOA systems are generally implemented using Web services standards, and these standards are agnostic of hardware platforms, software platforms or programming languages. Using Web services standards is not the only way of developing SOA applications, a different technology can also be used to implement SOA systems.

SOA has become a critical component in the development of enterprises that are more streamlined and better coordinated. SOA serves the purpose of greatly reducing costs while providing more efficient systems. It seems clear that today’s organizations will have to adapt such architectures if they want to remain competitive as noted by Schulte and Natis [5].

1.2 SOA Related Terms

Some important SOA related terms are given in the following subsections:

1.2.1 Service

The major pillar of SOA is the service. Services are coarse grained pieces of logic that are published by service providers and invoked by service consumers. A service provider exposes contracts for services and all the functionality exposed in the service contract should be implemented by the service. Services are discussed in more detail in Chapter 2 of the thesis.

1.2.2 Service provider

Service providers according to SOA are systems or programs that exposes services. It is service provider's responsibility to publish service contracts, and make sure that every exposed service implements all the functions (or operations, in SOA terminology) that are defined in its service contract. In the enterpise world, service providers can scale from small applications that can publish group of functionality to the bigger programs or systems as utility services to massive enterprise systems whose services enable EAI (Enterprise Application Integration).

1.2.3 Service consumer

In SOA, service providers publish services, and service consumers invoke the services' functions by sending and receiving messages to/from service providers. A service consumer can be any software that interacts with the service provider in order to invoke service operations. Client applications, other enterprise systems, even other services belonging to the same corporation can be a service consumer.

1.2.4 Contract

Service's contract corresponds to the collection of all the messages supported by the service collectively. The service contract is published by the service provider, and service consumers implement the logic for calling the service using this contract. In this sense, the contract constitutes the interface of the Service similar to interface of

an object in object oriented languages. In Web services, service contract is encapsulated in the document the service provider supplies called WSDL (Web Services Definition Language). The consumers get the WSDL document and implement code for communicating with the service prior to calling provider's service functions.

1.2.5 End point

An endpoint is a URI (Uniform Resource Identifier), or an address via which a service can be communicated. In Web services, service contracts, that is WSDL documents, contain endpoints to communicate with the service. Each endpoint can support different transfer protocol or message format.

1.2.6 Message

Messages are the units of communication in SOA. In a request/reply semantics for instance, a service consumer sends a message to the service provider in order to invoke a specific service function. The service provider parses the message, invokes the service operation, and returns the result as another message to the service consumer. Messages can have different formats (i.e. SOAP messages, HTTP GET messages etc.) and they can be sent using different transport protocols (i.e. HTTP, JMS, SMTP). In nowaday's Web services implementations for instance, messages rely on the SOAP standard. SOAP messages have a header and a body. The header does not contain the parameters related to the service function to be called or service reply, but general information that is required for tasks such as routing messages, handle authentication, to be used by the infrastructure that enables SOA.

1.3 A Brief History of SOA

The term SOA is first defined by Schulte and Natis [4], Gartner analysts, in 1996. In their work called "'"Service Oriented" Architectures, Part 1", they defined SOA as:

"A service-oriented architecture is a style of multitier computing that helps organizations share logic and data among multiple applications and usage modes.".

As noted by Natis [5], integration of different applications at that time was very problematic. Different propriatery integration technologies were being used to accomplish such needs. Common Object Request Broker Architecture (CORBA), Distributed Component Object Model (DCOM), Electronic Data Interchange (EDI) and Remote Method Invocation (RMI) can be given as examples of such propriatery integration technologies. These were successful at exchanging data between different applications, but they were propriatery. DCOM for example, is specific to the Windows applications, RMI on the other hand, is specific to the Java language. Some others, such as CORBA, were very complex to use, and were not widely adopted. Sometimes, in order to share data between different applications on different platforms, it was necessary to make these technologies communicate with each other, which was a very difficult task. ICT world was in need for a new perspective that could enable systems to communicate with each other regardless of their hardware and software platforms, regardless of programming language that they were implemented with. In short, application integration was the main driving force behind the appearance of SOA.

As noted by Erl [6], the service oriented architecture concept gained importance especially after the emergence of the web services technology. Erl summarizes the relation between these two technologies as follows:

“...Web services are about technology specifications, whereas SOA is a software design principle. Notably, Web services' WSDL is an SOA-suitable interface definition standard. This is where Web services and SOA fundamentally connect. Those who see Web services as architecture regard WSDL as the definitive standard of Web services (others see SOAP as a definitive standard for Web services — this is a view of Web services as a communication method). In practical use, the ubiquitous Web services standards enhance the mainstream appeal of SOA design.”

1.4 Promises of SOA

As noted by Roden and Lublinski [7], there are many business drivers associated with the decision to adopt SOA. These business drivers and the SOA solutions to these business needs include:

- *Restricted budgets*: SOA, if adopted properly, decreases development costs. SOA minimizes duplication of functionality between systems. It enables reuse of assets across multiple applications. It also decreases development costs, because it eliminates the need for software adapters functioning as a bridge used for integrating different software systems.

- *Restricted time*: SOA, if adopted properly, decreases development time and brings efficiency in terms of time to market, since the implementation of services can be done in parallel. If Web services technology is adopted for instance, many open source or commercial tools can be used to generate web service server application from the existing software code. Service proxy code to be used by the client applications can also be generated automatically using the service contracts. Developers can easily invoke the service operations using the generated proxy code[1], as if the service is part of their own system. This shields developers from the complex nature of networking and learning internals of the service provider. In this way, the time required for development decreases.

- *Rapidly changing business processes*: It is desirable to provide agile support to business processes, and to handle change management impacts more efficiently and effectively. SOA simplifies the impact of application upgrades. SOA brings agile support to business processes, handle change management impacts more efficiently and effectively. If the system is constructed using services, every service encapsulates a certain functionality. It is easy to spot the services that needs to be changed according to the changing business needs. Moreover, if Web services technology is used for instance, these changes can be done both on server side and on client side quickly, using tools that are able to generate server and client code automatically.

- *Speed and ease of project deployment:* SOA systems need servers where the services are deployed to be consumed by the service consumers. Many SOA

[1] Proxy code: Code which contains proxy object definition(s). Proxy objects mimic remote objects. When an operation of a proxy object is invoked, the proxy object in turn invokes the corresponding operation of the remote object that it represents and returns the result of the remote operation to the caller.

products, especially Web services systems make changing the service code and deploying the new service runtime to the associated servers very easy.

- *Developing technologies for identical business functions:* Since services can be regarded as free standing constructs, the same service installation can serve to multiple applications. SOA minimizes duplication of effort during development through reuse of existing services, enables efficiency in terms of time to market and development costs.

- *Need for integrating systems*: Systems to be integrated can be deployed in different geographies, run on diverse hardware platforms, implemented using different programming languages. For example, when a core banking system must be accessed and used by customers through the Internet, a demand is placed on the system to expose some of its functionality to the application that drives the Internet access. SOA enables true EAI (Enterprise Application Integration), brings diverse lines of business across many geographies together. SOA appears as a solution to many system integration related problems, as it enables systems to invoke various business functionality of other systems. In fact, this is the most important benefit of applying SOA.

- *Ease of development*: SOA eases development by shielding developers and integrators from system complexities. SOA is also beneficial for both developers and integrators because they no longer have to understand low-level systems in order to develop and manage complex processes. In addition, SOA allows applications can be managed and altered independently because low-level functions can be shielded behind high-level services.

- *Legacy systems integration*: A service can be developed in a higher level than an existing legacy system so that one face of the service can look at the modern world, speak the modern languages, such as SOAP. The other face of the system on the other hand, looks at the legacy system, and talks to the legacy system using its own propriatory language. In this way, using services as a bridge, it is possible to enable communication between modern world applications and legacy systems.

- *Ease of mapping business functionality onto ICT systems*: SOA helps in mapping business functionality onto ICT systems, since every service may encapsulate a specific business functionality. This is beneficial to businesses because it allows managers to have better insight into all of the processes of the business. This is possible because SOA defines the services that the managers require. In addition, it allows integration specialists, application managers, and other experts have control over the service implementation.

If the integration of SOA is always specific to a programming language or environment of an application, then integration effort will necessitate a great deal of custom integration work [3]. The capacity to link various systems together rapidly will require highly specialized programming capabilities and knowledge of the internals of these systems. In addition, ICT managers will be able to significantly reduce costs by using these general protocols and standards. These are the necessities that drive the importance of Web services technologies.

There is a theory that asserting that the redefining of infrastructures with a service oriented approach using standard interfaces between components should lead to benefits in the area of systems integration, and expert practitioners have confirmed that such benefits exist [8].

Since there are so many enterprises that are dealing with fragmented applications and databases, it is important that professional practitioners are focusing of the importance of SOA in ***integration of information***. In addition, the benefits associated with application development and maintenance and in particular the reduction of redundant functionality amongst systems along with the reduction of duplication during the development process, strengthens the position that SOA can have a positive impact on ICT service delivery [8]. In that article, it is further asserted that “It is also interesting to see the prominence given to more effective mapping of ICT systems onto business, which is another theoretical benefit of SOA. Clearly, it is much easier to map a collection of services onto business processes than it is to map functionality of a traditional monolithic application. The increased precision and flexibility in this is particularly relevant in a dynamic environment in which business processes frequently change.” [8].

It is also noted that, there is evidence suggesting that placing an emphasis on the service, which exceeds the systems and business worlds, can overcome the language problems that are often in the way of efficient communication between ICT and the enterprise [8].

According to Frievald [9], SOA was originally used as a way to reduce cost associated with integrating large business processes. Frievald[9] contends that the process was rather simple and involved “applications and automated processes access information resources through standard service interfaces, without requiring programming or knowledge of lower-level systems.” Web services provided the open standards required to employ an ever-present, reusable business function, which allow business processes to be separated and utilized as basic, controllable entities.

Executives are realizing the benefits associated with SOA especially in the past ten years. In fact, many executives believe that benefits promised by SOA make the implementation of SOA a must [10]. Frievald[9] further states that although there was some initial skepticism associated with the implementation of SOA, there now appears to be a great deal of success associated with it. According to a survey conducted in 2005 more than 50% of the respondents reported that they were familiar with SOA and 87% reported that they had a solid understanding of the construct of SOA. It is claimed in a white paper that such a rapid understanding of SOA is a proof that it is a strong construct and it has the capacity to produce many benefits [10].

2 SERVICES

Services are building blocks of SOA based applications. In this chapter, detailed definition of a service in the context of SOA is given, and characteristics of services are explained.

2.1 Definition of a Service

A service in the context of SOA can be defined as a software component that encapsulates a specific business function. A service is well-defined, self-contained, and it does not depend on the context or state of other services (i.e. services are loosely coupled constructs). A service should provide a high cohesion and a distinct function, should contain a coarse grained pieces of logic. A Service should implement at least all the functionality promised by the contract it exposes. One of the characteristics of services is service autonomy, which means that every service should be self-sufficient as noted by Rotem [1].

As noted earlier in Chapter 1, according to SOA, an enterprise application is composed of services. A business process for instance, is a process that encapsulates a business workflow that is composed by combining one or more services.

A service has a description or specification, and this description consists of an explicit and detailed narrative definition supported by a low-level (not implementation specific) process model. The narrative definition is in some cases augmented by machine-readable semantic information about the service which facilitates service mediation and consistency checking of enterprise architecture [11].

2.2 Characteristics of Services

According to Erl [12], there are some common service principles that are apparent with SOA. Erl underlines eight principles associated with SOA. In the following subsections these principles are summarized [12]:

2.2.1 Services share a formal contract

Service contracts play a significant role in SOA. Erl[13] asserts that, services are formally defined using one or more service description documents. As it relates to Web services, the technical service description documents usually include the WSDL service definition and the XSD (XML Schema Definition) schema. There is also a third type of document known as the policy that will grow in its importance over the course of SOA adaptation. All of the aforementioned documents may be categorized as service metadata, since each of the documents provides information concerning the service. In addition, Erl [13] asserts that the service description documents are viewed collectively as creating a service contract. In other words, the service contract is a series of terms and conditions that have to be both met and received by the service requestor to allow for successful communication and interfaces. The service contract can also be inclusive of other non-technical documents or legal agreements. Related to SOA, service contracts identify the formal definitions of the following:

- Service endpoints
- Service operations
- All of the input and output messages maintained by each operation
- The data representation model of each message's contents
- Regulations and characteristics of each service and the manner in which they function

Indeed shared service contracts are essential to the stability of any SOA initiative. Erl [13] further explains that “Service contracts therefore define a great deal of the underlying architecture of a solution environment, and may even provide semantic information that explains how services as part of this solution go about accomplishing a particular task. Either way, this information establishes the terms of engagement to which consumers of a service need to comply. Because this contract is shared amongst services, its design is extremely important. Service requestors that agree to this contract can become dependent on its definition. Therefore, contracts need to be carefully maintained and versioned after their initial release.”

Service contract is also important, since, in the Web services projects, the software code called **stub** on the client side, is generated from the WSDL document, which is

the service contract. When a WSDL document changes, all the clients depending on this contract should regenerate their stub code and change their service consume code accordingly, to remain compatible with the service provider.

This description of service contracts is an evidence that they are a chief component in the implementation of service oriented architectures and these contracts allow for the other seven principles including service abstraction, composability, and discoverability and others.

2.2.2 Services are loosely coupled

It is difficult to predict the manner in which the ICT environment of an enterprise will change over time. It is also impossible to predict the manner in which automation solutions will develop, integrate or replaced, and because of this, none of these changes can be planned. One of the primary purposes of SOA implementation is the ability to respond quickly to such unforeseen changes.

Coupling can be described as a measure of dependency. If the dependency between the software components is high, the coupling between them can be said tight or high. Two software components are said to be decoupled, if there is no dependency between them. In order for SOA to produce the aforementioned benefits, the relationship amongst the services must be loosely coupled as noted by Erl [13].

Implementing loose coupling is not only related to decreasing dependency between services. It also relates to decreasing dependency between a service and its consumers. This is accomplished by designing services that are not specific to specific service consumer. In this way, a SOA application may have a service inventory containing services encapsulating standalone units of logic. These services, as they are loosely coupled, can be used in different compositions.

In addition, as multiple services are being developed to satisfy the needs of a service oriented enterprise, such services may be organized into particular service models, which permit the services to be specialized in the same logic in which they encapsulate. The business service model is particularly critical as it relates to this construct because it restricts the logic that is represented by a service so that it reflects a correct expression of a particular business context.

When standardization occurs based on a set of service models, various domains of an organization have the capacity to be abstracted. This type of abstraction permits the organization to develop relationships that are loosely coupled amongst particular areas of the organization. This is accomplished when layers of services in SOA are developed that adequately correspond to the relevant domains' business functions. Once this is successful, loose coupling is beneficial because it improves the agility of the system. In short, it appears that loose coupling is an essential principle in SOA, because it allows more flexibility to the organization.

2.2.3 Services abstract underlying logic

This SOA principle concerns with abstracting functionality and technology. Erl [14] asserts that this phenomenon is also called service interface-level abstraction. This particular principle is responsible for encouraging enterprises and ICT professionals to create services as black boxes. These black boxes shield fundamental details from possible service consumers. This abstraction occurs via closely controlled usage of service contracts. This principle is necessary, because by restricting the public information to what has already been identified through the service contract, there is a greater level of separation between what is consumable and what is hidden. This principle related to abstraction is also essential as it relates to the principle of loose coupling.

As noted by Erl[14], there is no limit associated with the amount of logic that can be present in a service. For example, a service can be developed to carry out a simple task or can be formulated as an interface to a complete automated solution. In addition, no limitations are associated with the supply of automation logic that a service abstracts from. This means that coarse grained services can have the capacity to render logic from a multiple underlying systems. Even these systems may reside in different platforms, implemented with different programming languages. As organizations and ICT professionals evolve to the usage of service models that create a context related to a business unit or task, it is probable that in environments with various legacy solutions, one service will commonly render functionality that is dependent upon various systems. This hides details of the various underlying systems from the service consumers, since consumers do not need to communicate with the different proprietary technologies.

Abstraction at the interface level is a benefit of a distributed platform that is present with component and web services based architectures. In addition, the utilization of Web services particularly increase system-wise business relationships because they increase the amount of abstraction. Web services are also responsible for abstracting proprietary implementation details. This type of abstraction prevents users of the service from being forced to interface with the proprietary technologies of a particular vendor. Although abstraction is usually associated with a particular service, separate operations packaged in a service are actually responsible for abstracting the underlying logic. From this perspective, services can be regarded as vessels for the operations to carry out tasks. As a result of this, the amount of abstraction in a service can be identified by the combined levels of abstraction accomplished by each of the service operations.

Erl[14] claims that this particular principle places a great deal of emphasis on the manner in which the service contract is designed. If the service contract is generic in its design, the less consumer-specific the service will be. This issue determines reuse probability of what is rendered via the service contract.

2.2.4 Services are composable

A service can be composed of other services. As service portfolios increase in their capacity, service compositions became frequent in service oriented architectures as noted by Erl[15]. This principle is critical because it guarantees that services are developed in a way that will allow them to be members or controllers of compositions.

Development of the service functions is important for the service to be composable. It is important to understand that the composability of a service is actually a type of reuse and as such functions must be developed in a way that is standardized and that contain the right amount of granularity, so that advantage of composition opportunities can be taken.

A universal SOA principle that emphasizes the importance of composability is orchestration. As it relates to orchestration, a service-oriented business process is articulated through a composition language, such as BPEL (Business Process Execution Language). This composition language serves the purpose of organizing

the business process as a service composition embodied by a parent process service. In either case, the necessity for a service to be extremely composable is not contingent upon existing composition requirements.

2.2.5 Services are reusable

A service oriented architecture supports reuse and this creates flexibility and decreases time to develop an enterprise system. Main goal of the reuse principle is to develop every service as an asset of ICT that has repeatable value. When the number of these reusable assets increase, the chances increase that the effort to create new business automation requirements by building less and using more of what is already present will be successful.

This particular principle is geared toward reducing the time required to develop automation logic. In doing so, the overall responsiveness to change will improve for the enterprise, and the system becomes more agile. Additionally, the completion of development requirements will also become more cost-effective. Many enterprises are investing a great deal in the development of a reusable service inventory to accomplish these benefits. As noted by Erl [14],

“This principle facilitates all forms of reuse, including inter-application interoperability, composition and the creation of cross-cutting or utility services. As we established earlier, a service is simply a collection of related operations. It is therefore the logic encapsulated by the individual operations that must be deemed reusable to warrant representation as a reusable service. The emphasis on far reaching reuse also highlights the suitability of Web services as an implementation option for services. By making each service available via an industry standard communications framework, reuse potential can broaden dramatically because the logic encapsulated by a service now becomes accessible to service requestors built with different underlying technologies.”

It is apparent that the principle of reuse is important in realizing the full benefits of SOA. Reuse allows for the streamlining of the ICT environment, making it more cost effective and more efficient.

2.2.6 Services are autonomous

Autonomy is a critical principle associated with the implementation of SOA. If the performance of services is to be reliable and predictable, they must have a high level of control over the underlying resources. The principle of autonomy is consistent with this measure, as it stresses the necessity for individual services to have significant amounts of individual autonomy.

Such autonomy is necessary because increasing the level of control a service possesses over its implementation, the greater reduction in dependencies it will require, as it relates to shared resources within an organization. This makes a service more free-standing and reusable. Although a service cannot always be provided with exclusive ownership of the logic present within it (for instance, a service operation may need to perform database operations, and database may not be available when the service needs to connect to the database), the main concern of the enterprise is acquiring significant amount of control over the logic that is represented when the service it is executed.

Poulin[16] defines three types of autonomy:

a. *Pure autonomy*: Pure autonomy means that the service owns and controls all used components and resources. It is usually more desirable for an organization having a service inventory that contains pure autonomy, but it is a very rare case in real life. Erl [17] states that pure autonomy assists the organization in confronting scalability issues such as concurrent access conditions. Pure autonomy empowers enterprises to position services in a way that is more reliable to counteract the single point of failure possibility that often occurs as a result of leveraging reusable automation logic. However, because such autonomy commonly requires the development of new service logic and may demand special deployment conditions, it may produce further expense and effort.

b. *Service- level autonomy*: In this case, the service does not own and control some of its resources. This is the most common scenario in real life. As it relates to service-level autonomy, the limits of services are different from one another, however more than one service can still share the same resources.

For instance, Earl[17] states that “a wrapper service that encapsulates a legacy environment that also is used independently from the service has service-level autonomy. It governs the legacy system, but also shares resources with other legacy clients.”. In addition, pure autonomy asserts that the underlying logic is subject to the power and possession of the service. According to Erl[17], this is usually true when the logic is developed specifically in support of the service.

c. *Contractual autonomy*: It means that a service’s autonomy is based on service contracts with others.

Poulin[16] states that “One of the most dangerous things we have to avoid when dealing with service autonomy is the absence of control over the used resources. For example, if we use the service-level autonomy where the service, e.g., consumes data from a shared data-store, the mandatory element of such service design is the contract with the data-store. The contract has to define minimal but guaranteed number of concurrent connections available to the service on-demand, a range of acceptable response time values, schedule of data-store availability and a like. If the service design, on the contrary, assumes usage of the data-store silently, without a contract, it risks sporadic failures caused by potential mismatch between the service needs and data-store capabilities.”

2.2.7 Services are stateless

State describes the condition of being for an entity. For instance a car that is being driven is in a state of motion, as opposed to the parked car which is in a stationary state. Software components can be stateless and stateful. Erl[17] insists that these two terms define the runtime condition of a software program as it coincides with the processing necessary to perform a particular task. The automation of a specific task requires the processing of data that is related to that task. This data is known as state information. Software that is actively processing state information is identified as stateful.

The increases of the processing demands on services that are reused and composed also increases the necessity to take full advantage of the service processing logic. The development phase of service oriented architectures calls for a great deal of

emphasis to be placed on state management. As a result of this emphasis that is placed on the reorganization in the management of state information, SOA requires that there should be a principle that is designed to address these concerns.

This principle asserts that services should reduce the amount of state information they control, in addition to the length of time for which they remain stateful. Within an architecture that is service oriented, state information is usually representative of information that specifically relates to the current service activity. For instance, if a service is processing a message, its stateful condition is only momentary. In addition, if the stateful condition is long term, it will not have the capacity to be available to concurrent consumers.

SOA prefers a condition of statelessness as it relates to the services. The statelessness encourages autonomy, scalability and reusability.

2.2.8 Services are discoverable

The principle of discoverability is important because it aids in avoiding the creation of services that are redundant. It also prevents the implementation of services that contain redundant logic. Since each service within SOA architecture has automated logic that may be reused, the metadata linked to a service must have the capacity to accurately define both the functionality supplied by the individual operation of the service and the purpose of the service.

Erl [15] explains that the principle of discoverability is related in some way to architectural level discoverability, but it is also distinctive from this type of discoverability. As it relates to the architectural level, discoverability refers to the technology architecture's capacity to offer a discovery mechanism, such as a service registry. Such mechanisms effectively are a part of the overall infrastructure that supports SOA. Even at times when there is not a necessity for a service registry, services within the architecture must still be developed as highly discoverable. In doing so, when a service portfolio increases in size, the new type of management for those services can be better controlled, since each service possesses sufficient metadata to correctly communicate its purpose and capacities.

After reviewing the eight principles given above, Erl [12] states that “Of these eight, autonomy, loose coupling, abstraction, and the need for a formal contract can be considered core principles that form a baseline foundation for SOA. Although each of the eight principles support or are supported by the others, these four directly enable the realization of the remaining principles. It is also worth noting that Web services naturally support a subset of these principles, which provides an indication as to why the Web services technology platform is considered so suitable for building service-oriented solutions.”

All of these principles assist in the implementation and management of SOA within the enterprise. It is evident that organizations must have a full and thorough understanding of these principle prior to implementing a SOA application. If these principles are not thoroughly understood, an enterprise could waste a great deal of time and money implementing a SOA plan that will be unsuccessful.

3 WEB SERVICES

Web services technology have taken the concept of services and implemented it as services delivered over the web using technologies such as XML, Web Services Description Language (WSDL), Simple Object Access Protocol (SOAP), and Universal Description, Discovery and Integration(UDDI). Nowadays, web services is the primary technology that is used in implementation of SOA based applications. Using web services, it is possible to integrate different applications even if they are written in different programming languages, work on different operating systems and deployed on different hardware platforms. SOA, powered by web services technology, is the premier integration and architecture framework in today's complex and heterogeneous computing environment. In this chapter, web services is defined, and a typical scenario incorporating web services is given. Advantages and disadvantages of web services technology are also underlined.

3.1 Definition of Web Services

The software industry understood that combining software applications across numerous programming languages, operating systems, and hardware platforms leads to the problem of application integration. This problem is not going to be remedied through the use of a single proprietary environment and communication technique.

IBM defines Web services as a technology that permits applications to communicate with one another utilizing a platform and programming language in a manner that is autonomous [18]. Additionally, a Web service can be a software interface that depicts a collection of functions that can be made available throughout the network using standardized XML messaging. This approach utilizes protocols derived from the XML language to define a function that is being deployed in a system, so that data can be traded with other systems or some other Web service. In other words, Web services are software components capable of communicating with each other over multiple networks using universally accepted, nonproprietary open-standards based on XML. Overall, the most important problems that Web services attempt to solve are the problems of data and application integration.

Web services utilize XML, which has the ability to identify all data in a manner that is platform-independent. This allows for exchange across numerous systems, allowing for applications that are more loosely-coupled. In addition, Web services can operate at an intensity that is more abstract in that it has the capacity to reevaluate, alter or handle data types vigorously and according to demand. As a result, Web services are able to manage data more easily and permit software to liberally communicate [18].

At the conceptual level, Web services can be viewed as units of work, with each unit managing a particular task, containing group of operations related to that task. These operations can be integrated into business-oriented tasks to manage specific business operational tasks, and as a result, non-technical people can develop applications that can manage enterprise-wide issues collectively, utilizing a workflow of Web services applications. When the Web services are developed and built by technical professionals, business process architects can understand them, so that they can resolve business level problems. For example, an architect of a business process can attempt to compile an entire car engine using the car frame, body, transmission, and other systems, instead of looking at the various components contained within the engine. Likewise, this unique platform means that the engine can work in congruence with the components from different car manufacturers [18].

This scenario is analogous of the fact that Web services are assisting in bridging the gap between the business staff and ICT staff in an organization. The gap is bridged because Web services simplify technical operations for the business people to comprehend them more efficiently. As a result, business people can now detail events and actions and the ICT staff can associate these events and actions with appropriate services.

In addition, universally identified interfaces along with well designed tasks make it easier to reuse these tasks. Reusability of application software is important because it translates to an improved return on investment on software, as it can generate more from the same amount of resources. This also permits business people to consider utilizing an existing application in a different manner or even presenting the existing application to a partner in a different manner, and as a result, the business transactions between partners are enhanced.

According to an article published by IBM, web services are the primary technology that is used in the implementation of SOA [18]. Indeed nowadays, both XML and Web services play a significant role in businesses and SOA implementations. Although web services are widely used, there are also other technologies that can be utilized in the implementation of certain SOA components. Most SOA initiatives that involve Web services include the incorporation of legacy data that is present in technologies including Common Object Request Broker Architecture (CORBA). Businesses can execute SOA by simply using CORBA, Microsoft's Distributed Component Object Model (DCOM), or Remote Procedure Call (RPC) technologies.

3.2 A Typical Web Services Scenario

Figure 1 depicts a typical web services scenario:

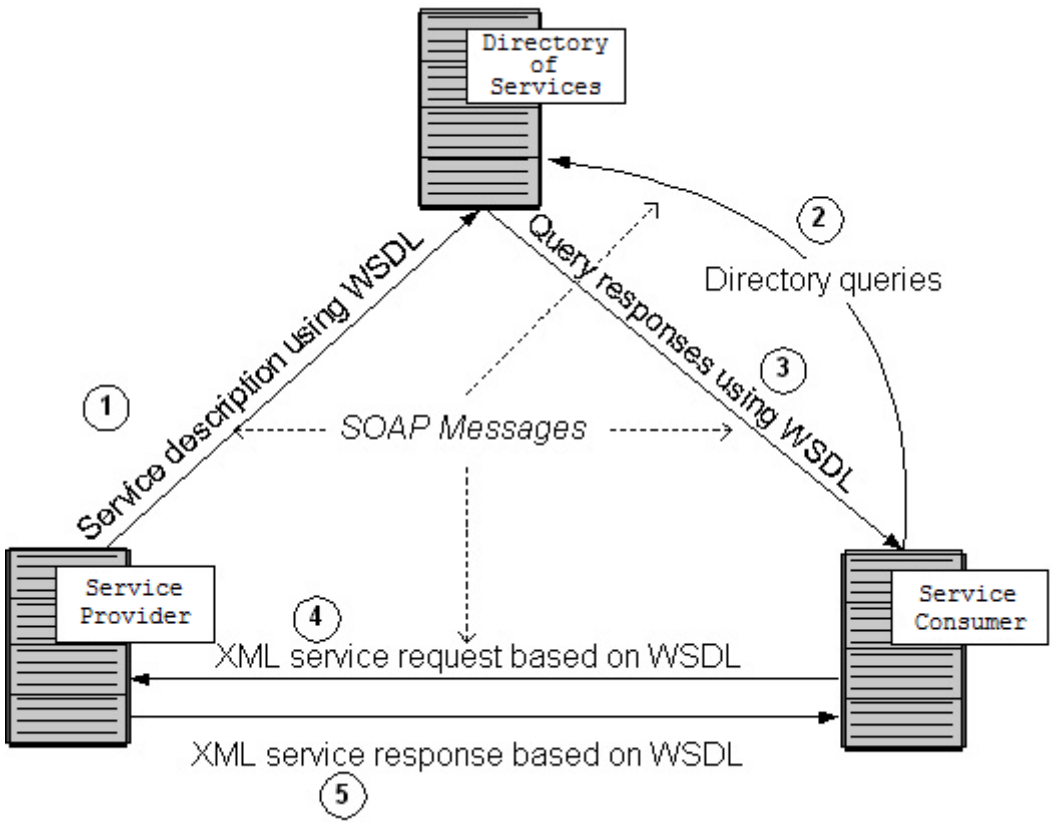

Figure 1. Web services scenario

The Web Services Description Language (WSDL) forms the basis for Web Services. A WSDL file is published by a service provider and contains information regarding every operation that a web service embodies. A service provider and service consumer are depicted in Figure 1. The steps involved in providing and consuming a service are:

1. A service provider describes its service using WSDL. This definition is published to a directory of services. The directory could use Universal Description, Discovery, and Integration (UDDI) protocol. Other forms of directories can also be used.

2. A service consumer issues one or more queries to the directory to locate a service and determine how to communicate with that service.

3. WSDL provided by the service provider is passed to the service consumer. This tells the service consumer what operations are provided by the web service and how to call them. It basicaly contains details of requests and responses for the service provider.

4. The service consumer uses the WSDL to implement a stub that is used to send a request to the service provider. All the request and response messages in Web Services technology are SOAP (Simple Object Access Protocol) messages, which is an XML based message format.

5. The service provider provides the expected response to the stub and and this response is passed to the service consumer.

Note that steps 1 to 3 are optional. A service consumer may already know the service provider and it can request WSDL related to the web service it would like to use directly from the service provider or get it by other means. Once the service consumer gets the WSDL document, it has all the required information regarding reaching the service and using its operations.

3.3 Characteristics of Web Services

Web-services technology has advantages and disadvantages. It is important to take into account all its limitations.

Advantages of Web-Services are noted by Shailesh[19] as follows:

- Web services are open, international standards-based.
- Web services are platform and technology neutral.
- Web services promote reusability.
- Web services make true EAI (Enterprise Application Integration) possible.
- Web services reduce integration costs. Web services allow companies integrate their own business processes with the business processes of business partners and clients, with the smaller cost than by using other integration technologies.
- Since Web Services are organized into the public repositories (UDDI lists, ebXML lists or others), they are accessible to the interested people worldwide. As a result of this, the threshold of the exposure of companies for the new markets is lowered, whereas possibilities for the growth of client base on the contrary grow.
- Web services do not replace existing applications and components but they build a superstructure themselves above existing applications. Thus, the safety of the already made investments into ICT infrastructure is ensured.
- The construction of the new solutions by implementing Web services makes it easier for companies to make their solution fast to market and in cheaper manner, since Web services enable less technical business people to "assemble" software solutions without the need for coding.
- Web services allow for a more scalable architecture.

Disadvantages of Web- services are summarized as follows referring to Shailesh[19]:

- Web services standards may evolve. This means, Web services applications should continually adapt themselves to emerging Web services standards to keep place in the cutting edge technology frontlines.

- Some vendor solutions create single-vendor approaches which conflict with the open standards based vision of Web services.

- By utilizing HTTP, Web services can evade existing firewall security measures whose rules are intended to block or audit communication between programs on either side of the firewall.

- Web services may suffer from poor performance compared to other distributed computing approaches such as RMI, CORBA, or DCOM. This is a common trade-off when choosing verbose text-based formats. XML explicitly does not count among its design goals either conciseness of encoding or efficiency of parsing**.**

4 ISSUES RELATED TO THE IMPLEMENTATION AND MANAGEMENT OF SOA

As it relates to the implementation and the overall management of service oriented architecture, there are several issues that organizations must take into consideration. The consideration of these issues is critical to ensure that the implementation and management of the SOA is successful.

4.1. Approaches to Integration

As noted by Frievald [9], the manner in which enterprises choose to integrate SOA will greatly effect the benefits that are arrived from its integration. There are a variety of platforms that enterprises can choose from as it relates to the integration of SOA. Frievald asserts that one of the first things that an organization needs to consider when integrating SOA is functionality that is required to develop the SOA. These functionalities are summarized in the following subsections:

4.1.1 Intelligent routing

This functionality is related to service calls in addition to other events that allow collaboration by steering action in applications and other divisions of the enterprise. These factors require a tool for distinguishing such things as messages types, knowing the possible endpoints for messages, identifying data values in them, and steering messages to where they're needed. All of them must be taken care of without weighing-down developers with the details of communication instruments or application manipulations as noted by Patrick[20].

4.1.2 Transformation

A highly developed transformation engines are necessary because application must work together to offer services or take action against events that may occur. In addition to the transformation engine, providing mapping amongst the various representations of business entities is a critical part of application. Even though XML transformations are also critical, the ideal transformation engines can also handle non-XML formats such as EDI, SAP IDocs and flat files [20].

4.1.3 Tools for service monitoring

Because SOA has a more distributed architecture, organization must have the capacity to view who is utilizing their services and if service-level agreements are being obeyed [20].

4.1.4 Advanced capabilities

Certain capabilities such as allowing basic enterprise searches are essential because certain kinds of services fit more successfully into the distributed SOA architecture and helps extending services to partners and [20].

4.1.5 State management

Even though many experts agree on that services in an SOA ought to be stateless, long-running business processes that utilize such services also necessitate a state management capability such as the capability offered by a Business Process Execution Language (BPEL) engine [20].

As was mentioned previously, a service oriented architecture is representative of a consistent way to determine and access distributed services that invoke functionality which creates the preferred impact with measurable preconditions and opportunities. The services shield implementation details but have related service descriptions that present enough information to be comprehended by both the technical and business requirements for utilizing the service. In addition, the decision to utilize a service is usually dependent upon an understanding of the requirements and conforming to the requirements [20].

4.2 Approaches to Service Enablement

This section mentiones some important and commonly used software engineering techniques for enabling services.

4.2.1 Service proxies

Patrick [20] noted that, as it relates to the enablement of services, service proxies play a valuable role. Service proxies are the most widespread type of service enablement, they take on this role because there is often a lack of suitable

integration points between the information sources. Service proxies are responsible for creating an external wrapper. This external wrapper is installed in front of an information source and behaves as an entryway or "proxy" to allow admission to the information source. In addition, a service proxy for the purpose of accessing files in a file system can be developed to gain admittance to a file that is located anywhere in a file system. The service proxy can also be limited to only certain areas of the file system.

In addition to the aforementioned functions of a service proxy, they can also be used to shield the particulars of the costumers' actions as it relates to having a session with the information service [20]. This hiding or shielding of details also relates to altering the service request into a suitable set of interactions so that the information service may be updated or retrieved. It also allows for the formation of the information that will be given back to the customer. There are also more advanced situations that require the use of service proxies. In such cases the proxy has the capacity to shield the additional steps that are needed to identify the result that will be given back to the consumer. Such an advance situation would encompass the retrieval of additional meta-data, including data sensitivity labels that are in another information store and must be retrieved and assessed prior to being returned to the consumer.

Service proxies can be utilized to improve performance. This improvement is realized when proxies cache remote references and information. When this takes place the subsequent service call will not necessitate any further registry calls [20]. In addition, as noted by McGovern and Stevens [21], because the service contracts are stored locally, the consumer is able to limit the amount of network hops needed to employ the services. The article also asserts that proxies work to improve performance by removing the need for network calls because the functions are performed locally.

The article [21] further asserts that the proxy will allow local implementations of entire service methods which do not necessitate service data. Furthermore, methods such as calculations and currency conversions can also be performed within the proxy. In addition, even if the method necessitates the need for some service data, the proxy has the capacity to download the data one time and reuse the data for other method calls. When this technique is utilized, it is imperative that the proxy support only methods the service itself provides [20].

There are also some definitive facts concerning the design pattern of the proxy. It has been stated that the proxy serves nothing more than a local reference to a remote object. In addition, if for some reason the proxy alters the interface of the remote service, then technically, it ceases to be a proxy. Further, a service provider will make available proxies for several diverse environments [21].

Additionally, a service proxy is developed in the native language of the service consumer. For example, a service provider will provide proxies for Java, C#, Objective C, Visual Basic, and Delphi if they are the most probable platforms for the service consumer. Even though the service proxy is not necessary, it does have the capacity to greatly enhance expediency and performance for service [21].

4.2.2 Service adapters

As noted by Patrick [20], another critical component in service enablement is the use of service adapters. Service adapters are actually an embedded component of a service proxy. However, there are some concrete differences as it relates to the manner in which these approaches enable service. The primary differentiation can be made in that the service adapter is incorporated into the response processing logic of an information source. In the most common form of this incorporation can be seen in application packages or an application server that host the application, as opposed to an external component of the information source.

There is a primary benefit associated with the use of a service adaptor [20]. This benefit or advantage has to do with the fact that service adapters usually do not require another linkage between the information source and the service enablement piece. This is different from the service proxy approach in that there is usually a network link between the proxy and the consumer, and a connection between the information source and the proxy; however the service adaptor approach only necessitates support for the link between the adapter and the consumer. The benefit of this is that it usually produces an environment that is more secure because the communication with the information source is arbitrated by the adapter which is integrated into the request processing logic of the information source.

4.2.3 Unified information views

One of the critical issues that organizations must take into consideration while realizing SOA is unified information views. Simply revealing information sources as services does not take into account the issue of unified information views. Revealing information sources as services also has no effect upon the issue of updating the information that is contained in the unified view. Indeed a service based approach is problematic because it lacks the logic that is responsible for understanding the information sources that possess the information that is required to develop a unified view of a unit and the interactions between the assorted pieces of information found in each information source that compile the unified view [20].

Once an organization understands this dynamic a developer should organize an information service that will reflect a unified view [20]. Patrick concedes that this would be a difficult task because the developer would have to have knowledge of all of the information sources. This task is even more difficult when there is a requirement to carry out updates on the information that is within the unified view. In such a case, a developer might have to recall where the information was derived. The developer would then be forced to manage the problem associated with knowing what aspect of the information was altered that the related information sources that had to the updates. Additionally, the developer would have to confront the issue of not being able to update some of the data store [20].

Although this can be a significant issue there are products that address the problems associated with developing a unified view. These products as noted by Patrick [20], “provide service based views to application developers without the need to develop the actual code to assemble the unified view. Instead, these types of products utilize graphical modeling tools which can discover and display schema contained in structured storage and allow unified views to be graphically constructed and then code generated. Furthermore, the generated "code" can be in the form of a declarative query language, such as XQuery. This enabled a variety of optimization techniques to be brought to bear when processing service requests, particularly if the services are defined in a layered manner. These types of products also include features which handle the issue of utilizing the appropriate authentication mechanism

and connection management of each information source so that the identity of the requesting entity is available to the information source."

4.2.4 Data sources

There are many different sources of information within an organization which are inclusive of directories, images, text files and databases. In addition, the manner in which this information can be stored is also varied. In order to confront the problems that may arise as a result of all the varied information, a developer has to code to meet the needs of every source. In addition, as the information source grows and additional information sources are apparent, there must be changes made that adjust the specifics of the format of the information source. Even though SQL offers a generic purpose mechanism that allows access to the data that is located in databases, however the implementation of SQL as a way to get information from other information sources was never utilized. As a result, a concept arose involving the use of information sources as services [20].

In the use of Service Oriented Architecture, organizations are now depending upon service-based information sources. Even though this approach could be carried out with many different types of technologies, Web services and XML are the preferred technologies. Using SOA, organizations now have the capacity to hide details concerning whether the information source uses an SQL query or any other mechanism to gain access to the information [20].

As a result of this, technology developers do not have to learn several different access schemes. Instead developers will be able to focus on creating service requests and managing service responses. When organizations use certain technologies, developers can be taken away from the details of authenticating the information source to dealing with connection management.

4.2.5 Access and security

Patrick [20], asserts that information contained in a SOA is only as good as the security that the business implements. In most cases vendors tend to focus on certain aspects of security such as authorization, authentication and secure communication. However, the business environment also requires that other security

issues be considered. These additional security issues concern such things as new privacy laws and regulatory statues. In addition, there are now other business requirements that are an important component for most businesses. Making sure these security features are present is no longer an option as there are now serious personal and financial penalties that can occur if these new policies are not adhered to.

Contemporary business systems gather information from many diverse sources and this information is retrieved in various formats. With this in mind, it becomes an increasingly difficult task to manage all of these various forms of information in one type in information warehouse. Indeed businesses are forced to choose between many different security solutions that are not integrated and do not have the capacity to be managed from one location [20].

Indeed integrating SOA with the security of the organization can be a difficult challenge. However for many of the reasons that is previously mentioned, the integration of security is critical for the proper implementation and management of SOA. Integrating security is particularly difficult as it relates to SOA because there is a requirement that all the components of each application have a separate security administration. Initially businesses undertook these measures because it was believed that each component was vulnerable to security breaches. So then if information sources are taken into consideration it is apparent that every source has an authorization mechanism, an authentication mechanism and an auditing mechanism. Since all these mechanisms exist a great deal of money can be spent to manage all of these individual mechanisms [20].

Viewing this issue as a problem for businesses, some security vendors have attempted to resolve this issue via the concept of centralized enforcement servers. Many of the information vendors nowadays actually use a security system as the authority on policy decisions. However, Patrick [20] argues that large percentage of these systems has not been very effective. The effectiveness of these systems is further reduced if they are utilized in an SOA which is extremely distributed. For instance, the Department of Defense's Network Centric Enterprise Services (NCES), found that the network latency alone as it related to producing remote authorization decisions ended in a 70% performance deficiency for each web service business

request [20]. According to Patrick [20], the addition of extra bandwidth could combat many of the performance related problems. However additional bandwidth will not confront the network latency problems, or the potential for the security services to be a bottleneck or single point of failure.

As noted by Patrick [20], a common approach to security in the world of business is to protect the resource in a way that is a close to the resource that is possible. However, businesses must confront the issue of unifying the protection without degrading the performance. For this reason, there are certain products that offer a collection of security services that permit security measures to take place as close to the resources as possible. As with any other service, the execution details of involving the security service are abstracted via a service interface. In addition, the security constructs present in this approach may be used with the service proxy and service adapter approaches discussed earlier in the research; it can also be used within applications themselves.

Patrick [20] states that, to effectively implement a distributed environment there must be a centralized administration approach. Such an approach must also be established on services as to ensure that varied implementations can be incorporated permitting existing investments to be integrated. In addition, promising standards concerning provisioning and policy language, including the XML Access Control Markup Language (XACML) and Service Provisioning Markup Language (SPML) are the building blocks of the security mechanisms that are present in information sources and they have the ability to integrate.

4.3 Technologies Relevant to SOA

Two software technologies, namely ESB (Enterprise Service Bus) and BPM (Business Process Management) gained interest in the SOA world, since they simplify SOA implementations, improve their quality by adding new capabilities and increase their agility. They also reduce complexity of management of SOA deployments. These technologies are investigated in the following sections.

4.3.1 ESB (Enterprise Service Bus)

An ESB is a software system that functions as a transit layer for carrying information, providing connectivity to a wide range of heterogeneous systems. The bus provides a set of capabilities to enable application integration and service-oriented architecture (SOA), including service creation and mediation, routing, data transformation, and management of messages between endpoints. ESB is an important software product category today, many prominent software companies offer ESB products (such as Oracle ESB by Oracle, WebSphere ESB by IBM) and there are many open source ESB implementations (such as Apache ServiceMix, JBoss ESB, OpenESB, MuleESB, WSO2 ESB), since they act as integration platform that allows connecting various applications and systems together quickly and easily, enabling them to exchange data. Figure 2 depicts functions provided by a typical ESB from architectural point of view.

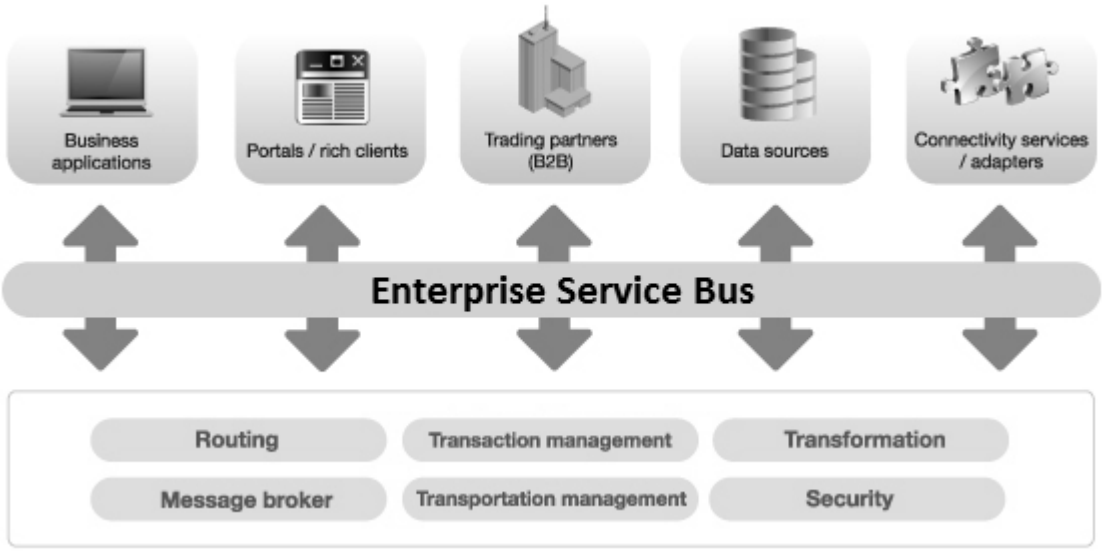

Figure 2. Enterprise Service Bus (ESB)

The key advantage of an ESB is that it allows different applications to communicate with each other by acting as a transit system for carrying data between applications within an enterprise or across the Internet.

Contemporary ESBs offer the following functionality:

- *Service creation and hosting*: ESBs expose and host reusable services, using Mule ESB as a lightweight service container.
- *Service mediation*: ESBs shield services from message formats and protocols, separate business logic from messaging, and enable location-independent service calls.
- *Message routing*: ESBs route, filter, aggregate, and re-sequence messages based on content and rules.
- *Data transformation*: ESBs exchange data across varying formats and transport protocols.

When an ESB is utilized, businesses have the capacity to rapidly put together applications from the services that are plugged into the bus and they can then develop new applications that are information based. The orchestration of services is now available to alter the manner in which the ICT infrastructure of a company uses information sources. It also changes the manner in which a company integrates the capabilities supplied by and to external partners to permit the business to efficiently execute its undertaking.

ESB is essential because it offers flexibility to permit data transformations and various services to be “plugged into the bus and then be reused by any number of different services and application components [20].” In addition, one of the primary benefits associated with the implementation of ESB is its ability to eliminate the need to make a particular processing flow hard code. As such, the processing flow can be defined and then assembled by non-developers.

Data transformations are a significant component in the implementation and management of service oriented architectures. The exposing of an information source as services does not give consideration to the concept of transforming information into a new format. Transforming information into a new format is often necessary because most enterprises have different identifiers associated with bits of information. Although it may seem beneficial to standardize the information models

associated with an application, in most cases it is neither possible nor practical. This is particularly true when there are aspects of an application are services supplied by an external partner. Consequently, there is often a requirement for data transformation even if information sources are offered as services [20].

There are many approaches that can be utilized to transform the data. The concept of canonical information format is an approach that can be used to rectify this problem. As it relates to the canonical approach, individual information services have to normalize information into a canonical format[2]. This particular approach does come with some problems including some limitations. The most prominent limitation has to do with the fact that many businesses do not have the ability to identify an individual data dictionary. In addition they do not have the capacity to guarantee that their partners will use the definitions that are identified [20].

Understanding this limitation, businesses may choose to embrace another approach that will allow them to develop other services that only offer the capacity for data transformation. When this type of construct is created, the business can then allow individual implementations to make requests to other services. Although this approach can be successful it does have some drawbacks. For instance the separation and reuse of data transformation can create hard coded process flows that might need to be altered [20].

In addition to the assertions made by Patrick [20], Arsanjani [23] states that businesses that have a variety of services will eventually need an administrative and operational model. Such a model will improve the ability to achieve the following:

- Maintain a range of interface techniques, such as messaging, synchronous request/response, and publish/subscribe.
- Manage a significant amount of large numbers of service interactions in a convenient manner.

[2] Canonical format can be defined in enterprise application integration context as, “the Canonical Model is a design pattern used to communicate between different data formats. It introduces an additional format, called the "canonical format", "canonical document type" or "canonical data model". Instead of writing translators between each and every format (with potential for a combinatorial explosion), it is sufficient just to write a translator between each format and the canonical format [22].”

- Offer support for highly developed service interaction capacity which would be inclusive of such things as store and forward transactions, infrastructure services, quality of service and security.

- Offer a strong, controllable, disseminated integration infrastructure that rely upon the aforementioned principles of SOA.

- Manage both Web services along with conventional EAI communication standards and technologies.

Arsanjani [23] asserts that a business that is confronted with such issues should develop an ESB that can supply middleware capabilities throughout the businesses so that all the various service interactions can be managed appropriately. Further, an ESB must have the capacity to support the mediation, transformation, integration and communications technologies that are needed by the services and capacity to distribute geographical deployment, utilizing a common model for both the management and administration of the enterprise. According to Arsanjani [23], the critical features of an ESB, include the following: “It is deployed as one or more "hub" components. Each hub provides a localized but resilient capability to perform routing, transformation, security, and other functions commonly referred to as "intermediary" functions. It has a namespace directory and administration services used to administer the services it supports access to. In a geographically deployed ESB, the administration services maintain a consistent configuration across a network of cooperating Hubs. It has a number of inbound ports. Each inbound port is configured to receive service requests on a set of addresses using a particular protocol, for example, SOAP/HTTP or WebSphere MQ. It has a number of outbound ports. Each outbound port is configured to forward service requests to and receive responses from a set of addresses using a particular protocol [23].”

This type of construct allows the ESB to support the virtual service provider pattern and remote service strategy for any number of services across an enterprise in a common way. In addition, the ESB can allow transformations between a variety of data format, security, or transactional models in addition to service requesters and providers. By possessing this function the ESB acts as both a control and

encapsulation point for the expected difficulty and different scenarios that are present in an organization.

Thus, the enterprise service bus is important in the development and implementation of an SOA. It is apparent that ESB assists businesses in streamlining various functions. It is also apparent that ESB is most useful as it relate to SOA involving large companies.

4.3.2 BPM (Business Process Management)

Noted by Smith [24], business processes are at the heart of business operations. Using SOA approach, software systems can be built using services. Like services, business processes can be considered as software blocks, but business processes tend to be bigger, much more coarsly grained constructions. It is natural for a business process to encapsulate a workflow containing calls to many SOA services deployed in geographically and technologically dispersed providers, some of which may belong to different constitutions. In business process world, the messages exchanged between different systems are business documents. These documents can be considered as parameters passed to service operations. In a way, it is possible to think about business processes as software blocks related to business, not with a specific product. Business process blocks perform workflows containing service calls to different products, and need to be managed separately from the services they connect to.

For decades business process management was limited to only the ability to document processes. As the means to document processes technologically improved so did the potential to use technology to manage more aspects of the process. Over the years built business process management tools failed to simplify a subset of the overall complex process like workflow, or enterprise application integration. Today, business process management products hold promise of a comprehensive solution to manage all enterprise processes with greater efficiency as noted by Smith[24].

SOA organizes ICT infrastructure to achieve greater efficiency through reuse, and better agility with the use of services. It is a response to proliferation in systems, technologies, communication protocols, databases and data models. SOA hides this complexity behind a set of business services and provides an infrastructure that

standardizes how these services communicate and how they are managed. The result is that ICT can respond faster and build new solutions for the business faster and cheaper. A transformation to SOA predominantly benefits ICT [24].

BPM and SOA, when combined work together to synergistically improve both business and ICT efficiency and agility providing higher order benefits. In addition, the unification of BPM with SOA delivers unprecedented simplicity and efficiency in a SOA environment [24]. Many leading software companies such as IBM and Oracle are aware of this and include a BPM implementation in their product catalogue.

4.3.2.1 Barriers to BPM

Business processes are by definition dynamic in structure as they manage operational flow in a business that must adapt to changing business conditions. It is challenging to manage these processes without incurring additional complexity. First generation solutions lacking completeness and minimal integration introduced barriers to business process management efficiency. Barriers included the inability to address all process types with one solution, products that add complexity by requiring ICT to first integrate the components of the BPM solution, ineffective collaboration capabilities that occur outside of the process context, and performance, scalability, and availability that failed to address enterprise requirements [24].

In the following, some of the barriers to BPM are outlined:

a. *Process diversity*: There is a natural diversity in the type of processes found in business operations. Processes may predominantly require human interaction, system to system exchange and transformation of information, the generation of documents, or the need to arrive at an important decision. In addition, processes can be characterized as simple or complex and may need to manage a limited or large number of transactions. Some processes, like a sales campaign, are long running and may take weeks, months, or even years while others occur in less than a second. Processes may also be characterized as highly structured with repeatable tasks or unstructured where the flow of the process can only be determined during process execution.

As noted by Harvey[25], the diversity of these processes has traditionally required the enterprise to use multiple process management systems that specialize in managing a subset of these process types. The unintended consequence results in trading process complexity for infrastructure complexity. Efficiency may be improved in some processes but overall efficiency and agility is still challenged due to the limitations or barriers imposed by a point process solution.

The ideal process management solution is a platform that is truly complete that it can address all types of processes. This reduces ICT complexity inherent in supporting multiple products. It also provides a long term growth path for managing more processes over time increasing overall process management benefits, return on investment, and lower ICT and business costs. This solution must also address the needs of both business and ICT with tools appropriate to each role.

b. *Collections instead of suites:* The component capabilities of business process management suites have existed separately for many years. Document management practiced in the 1980's demonstrated the value of workflow and human and process interaction. Business process management suites introduced the innovation of solution completeness with the inclusion of all required process management tools with the implicit promise that tools easily work together [24].

BPM's primary benefit is increased efficiency for processes that span multiple systems in the enterprise. Given this goal it makes sense that it is more efficient to start with a process management solution that benefits from a high level of internal integration simplifying the ability to remove complexity, deliver faster time to value, and improve business agility.

c. *Communication breakdown:* It is often said that business process management is essentially change management. One of the greatest challenges inherent in both is managing the human and organizational aspect of change. People naturally resist change for a variety of reasons. Successful

process management requires cooperation, communication, and collaboration of all key stakeholders in both the business and ICT [24].

Effective communication and collaboration within the cross-functional process team provides the foundation for success. Effective collaboration requires process context and trying to find a process notification in an ever lengthening list of emails provides a poor solution. In addition, the last few years has seen the emergence of the importance of social networking in the corporate environment. It is only natural that these new collaborative communication technologies have a place in bringing the organization together in the management of processes. Communication will benefit from the inclusion of new communication choices while process collaboration will be improved with communication in context [24].

d: *Enterprise Readiness:* Enterprise software requires enterprise-class capabilities and performance. As discussed earlier some processes require scaling to incredibly large numbers of transactions. Other processes may contain a complexity that requires the utmost in execution efficiency. In addition, like all enterprise software, availability, redundancy, and performance need to be provided for. Without an enterprise grade infrastructure, process management is constrained to singular departmental processes without stringent requirements for availability [24].

4.3.2.2 Properties of recent BPM software

Contemporary BPM products offer various features targeting to simplify BPM implementation and management. This section mentiones some important properties of recent BPM software.

a. *Unified process foundation:* As noted by Harvey [25], business process management has been shown to consistently simplify and increase efficiency for processes that connect isolated silos of information across the enterprise. It stands to reason that in order to help remove complexity business process management software should itself be unified.

Referring to Harvey [25], the unified process engines provide for modular plug-in execution of BPEL (Business Process Execution Language), the rules engine, and human workflow with common rules available across all modules. A unified engine delivers implementation of BPMN execution providing all the benefits designed in this latest standard. The business catalog is a common metadata store for both design and run time. End-to-end management unifies management and monitoring of business processes with the connecting middleware so that problem management does not stop at the process boundary. A problematic process can be inspected through its process state and into any interconnecting web services simplifying diagnostics and problem resolution.

Integration services simplify connecting to additional systems and include a variety of adapters for operational systems, integration with UDDI (Universal Description, Discovery and Integration) and WSIL (Web Services Inspection Language) repositories, imaging and content management systems, and the ability to integrate metrics from various operational data stores, messaging systems, OLTP systems, and process engines. It also unifies operation with various products such as business Intelligence systems [25].

Enterprise grade process management requires enterprise-class operational performance for both latency and throughput, high availability, and should be enterprise architecture capable. The unified process foundation provides enterprise grade support for both large numbers of processes and users. High availability from clustering and end-to-end high availability along with database high availability assures that processes will always be available.

b. *User-centric design:* There are many types of contributors to the process management lifecycle. They include process modelers, developers, approvers, and casual participants in both the business and ICT. Each role has specific requirements and each individual their own preferences. Recent BPM products address requirements for the many roles and preferences of users in both ICT and the business. They generatlly contain different tools for technical business analysts and similar varied roles of process participants in the business. Easy-to-use process analysis and report generation tools simplify process status

and operational detail reporting. Multichannel development tools enable simplified integration and reuse of interfaces across Web, portal, and service channels [25].

c. *Business process reports:* Process analysis and reporting provides critical business visibility. Business users require a wide variety of reports and the flexibility to tailor reports to changing requirements. Recent BPM tools provide easy-to-use, rich, visual report editor. Impact, gap, and redundancy and similar reports can be generated easily providing visualization of process efficiency and new simulation reports help visualize potential process scenarios [25].

5 AN APPLICATION

In this chapter, as an application of SOA, an IHE(Integrating Healthcare Enterprise) profile PDQ(Patient Demographics Query) is implemented using web services technology and deployed on an ESB acting as a service provider. Introduction to the IHE concepts and implementation details are given in the following sections.

5.1 IHE (Integrating the Health Enterprise)

One of the key problems in the contemporary eHealth systems is interoperability. As noted by Dogac et al. [26], interoperability concept in healthcare can be investigated in different domains, such as the interoperability of messages exchanged between healthcare systems, Electronic Health Records (EHRs), patient identifiers, coding terms, clinical guidelines and healthcare business processes.

The IHE Initiative is a long-term program sponsored by RSNA[3], HIMSS[4], ACC[5], ESR[6] and other professional societies to promote the coordinated use of healthcare ICT

[3] RSNA (Radiological Society of North America): "RSNA is a professional society established in 1915, working on improving patient care through education and research. More than 40,000 medical imaging professionals are members of RSNA, including radiologists, radiation oncologists, medical physicists and allied scientists.[27]"

[4] HIMMS (Healthcare Information and Management Systems Society): "HIMSS is a cause-based, not-for-profit organization exclusively focused on providing global leadership for the optimal use of information technology (IT) and management systems for the betterment of healthcare. Founded 50 years ago, HIMSS and its related organizations have offices in Chicago, Washington, DC, Brussels, Singapore, Leipzig, and other locations across the United States. HIMSS represents more than 30,000 individual members, of which two thirds work in healthcare provider, governmental and not-for-profit organizations. HIMSS also includes over 470 corporate members and more than 85 not-for-profit organizations that share our mission of transforming healthcare through the effective use of information technology and management systems. HIMSS frames and leads healthcare practices and public policy through its content expertise, professional development, and research initiatives designed to promote information and management systems' contributions to improving the quality, safety, access, and cost-effectiveness of patient care. [28]"

[5] ACC (American College of Cardiology): "The American College of Cardiology is transforming cardiovascular care and improving heart health through continuous quality improvement, patient-centered care, payment innovation and professionalism. The College is a 39,000-member nonprofit medical society comprised of physicians, nurses, nurse practitioners, physician assistants, pharmacists and practice managers, and bestows credentials upon cardiovascular specialists who meet its stringent qualifications. The College is a leader in the formulation of health policy, standards and guidelines, and is a staunch supporter of cardiovascular research. The ACC provides professional education and operates national registries for the measurement and improvement of quality care. [29]"

standards and to provide a common framework for multi-vendor systems integration. IHE re-uses existing standards such as DICOM[7] and HL7[8] as the building blocks for assembling larger integrated solutions, thus the IHE framework is not re-inventing a new standard in healthcare but provides a practical method to make these standards work.

High-quality patient care and optimal clinical workflow require efficient access to all relevant data in the healthcare enterprise. Integration of modalities and clinical ICT systems from a variety of manufacturers is essential to achieve this goal. In spite of integration solutions available from many manufacture, connecting systems from multiple vendors is a real challenge. IHE provides a common framework for building effective solutions to overcome integration problems.

The following are key problems of contemporary healthcare systems:

- Inadequate access to comprehensive medical information.
- Lack of consistency in medical data between departments and institutions.
- Errors in transferring medical data between systems from different suppliers.
- Risky investments in proprietary solutions instead of expandable open systems.
- Limitations in workflow optimization and inadequate support of user tasks.

---

[6] ESR (European Society of Radiology): "The European Society of Radiology (ESR) was founded in December 2005 by merging the European Congress of Radiology (ECR) and the European Association of Radiology (EAR), thus establishing a single house of radiology in Europe. It is an apolitical, non-profit organisation, exclusively and directly dedicated to promoting and coordinating the scientific, philanthropic, intellectual and professional activities of Radiology in all European countries. [30]"

[7] DICOM (Digital Imaging and Communications in Medicine): "DICOM is a global Information Technology standard that is used in virtually all hospitals worldwide. Its current structure, which was developed in 1993, is designed to ensure the interoperability of systems used to: Produce, Store, Display, Process, Send, Retrieve, Query or Print medical images and derived structured documents as well as to manage related workflow. [31]"

[8] HL7 (Health Level Seven International): "Founded in 1987, HL7 is a not-for-profit, ANSI accredited standards developing organization dedicated to providing a comprehensive framework and related standards for the exchange, integration, sharing, and retrieval of electronic health information that supports clinical practice and the management, delivery and evaluation of health services. HL7's 2,300+ members include approximately 500 corporate members who represent more than 90% of the information systems vendors serving healthcare. [32]"

IHE is working on defining the required interfaces to improve the exchange of data between Imaging and ICT Systems within and across healthcare units. It is a global initiative that creates frameworks for passing vital health information seamlessly from application to application, system to system, and setting to setting within and across healthcare enterprises.

IHE does not create standards, but creates technical frameworks and integration profiles. IHE promotes the coordinated use of established standards, such as:

- Healthcare content standards (such as HL7, HL7 CDA, DICOM).
- Electronic business standards (such as ebXml registry, SOAP, web services).
- Internet standards (such as HTTP, IETF and W3C).

The systems that conform to the IHE integration profiles have the following benefits:

- Efficient exchange of clinical information
- Easier systems implementations
- Workflow optimization
- Improved quality of patient care

IHE is supported by clinical users, administrators, ICT specialists, standards development organizations and the medical technology industry. IHE is an international endeavor, many countries have already joined the IHE initiative to adopt and promote the deployment of its integration framework in clinical practice at national level.

5.2 IHE Domains

IHE is organized for clinical and operational domains. In each domain users with clinical and operational experience identify integration and information sharing priorities and vendors of relevant information systems develop consensus, standards-based solutions to address them.

Each domain includes a technical committee, whose primary task is developing and documenting these solutions (known as integration profiles), and a planning

committee, whose primary tasks are long-term scope planning and organizing deployment activities (such as testing events and educational programs). Each domain develops and maintains its own set of Technical Framework documents.

Currently active IHE Domains include [33]:

- IT Infrastructure
- Cardiology
- Pharmacology
- Eye Care
- Laboratory
- Patient Care Coordination
- Patient Care Devices
- Quality, Research and Public Health
- Radiology
- Radiation Oncology
- Anatomic Pathology

5.3 IHE Profiles

IHE Integration Profiles describe the solution to a specific integration problem, and document the system roles (Actors), standards and design details for implementers to develop systems that cooperate to address that problem.

IHE Profiles are a convenient way for implementers and users to be sure they're talking about the same solution without having to restate the many technical details that ensure actual interoperability. For convenient reference, each Profile has a short acronym.

Each domain specifies a collection of Profiles for problems directly related to their domain. IT Infrastructure Domain, which is the central IHE domain related to healthcare organizations, contains the following active IHE profiles [34]:

a. *[ATNA] Audit Trail and Node Authentication*: authenticates systems using certificates and sends PHI-related audit events to a repository to help implement confidentiality policies.

b. *[BPPC] Basic Patient Privacy Consents*: records patient privacy consents, notes consent on electronically published documents and enforces privacy appropriate to the use.

c. *[CT] Consistent Time:* ensures system clocks and time stamps of computers in a network are well synchronized (median error less than 1 second).

d. *[XDM] Cross-enterprise Document Media Interchange*: transfers documents and metadata using CDs, USB memory, or email attachments.

e. *[XDR] Cross-enterprise Document Reliable Interchange*: exchanges health documents between health enterprises using a reliable point-to-point push network communication.

f. *[XDS] Cross Enterprise Document Sharing*: registers and shares electronic health record documents between healthcare enterprises, physician offices, clinics, acute care in-patient facilities and personal health records.

g. *[XDS-SD] Cross-enterprise Sharing of Scanned Documents*: couples legacy paper, film, and electronic documents with the healthcare metadata needed to manage them as electronic health record documents.

h. *[XUA] Cross-Enterprise User Assertion*: communicates claims about the identity of an authenticated principal (user, application, system...) across enterprise boundaries.

i. *[EUA] Enterprise User Authentication*: enables single sign-on by facilitating one name per user for participating devices and software.

j. *[PAM] Patient Administration Management*: establishes the continuity and integrity of patient data in and across acute care settings, as well as among ambulatory caregivers.

k. *[PDQ] Patient Demographics Query*: lets applications query a central patient information server and retrieve a patient's demographic and visit information.

l. *[PIX] Patient Identifier Cross Referencing*: cross-references patient identifiers between hospitals, sites, health information exchange networks, etc.

m. *[PSA] Patient Synchronized Application*: allows selection of a patient in one application to cause other applications on a workstation to tune to that same patient.

n. *[PWP] Personnel White Pages*: provides basic directory information on human workforce members to other workforce members and applications.

o. *[RID] Retrieve Information for Display*: provides simple (browser-based) read-only access to clinical information (e.g. allergies or lab results) located outside the user's current application.

5.4 Patient Demographics Query (PDQ) Profile

As defined by item (k) in the profile list above, PDQ (Patient Demographics Query) is IHE IT infrastructure integration profile that provides ways for multiple distributed applications to query a patient information server for a list of patients, based on user-defined search criteria, and retrieve a patient‘s demographic (and, optionally, visit or visit-related) information.

PDQ is an important profile, as it provides a way to obtain crucial demographics information related to patients to healthcare institutions. It is also important as various IHE profiles use this profile as a prerequisite. So, in a sense, it is used as one of the basic building blocks in creating healthcare systems relying on IHE profiles. For that reason, PDQ is selected to be implemented in this study as a SOA application. Also, this PDQ implementation can be used as an example in developing remaining IHE IT infrastructure profiles, as all IHE profiles are similarly constructed, and based on similar standards.

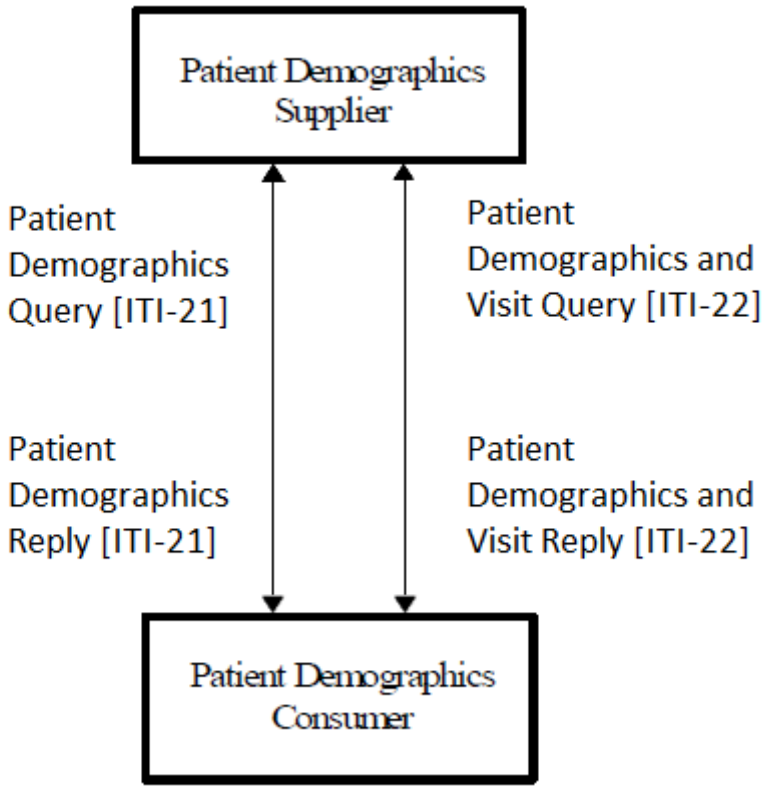

Figure 3. PDQ actors and transactions

5.4.1 PDQ roles

PDQ defines two actors, that represents two systems or software components that fulfills the roles assigned to them: Patient Demographics Supplier and Patient Demographics Consumer. Figure 3 depicts the actors directly involved in the Patient Demographics Query Integration Profile and the relevant transactions between them.

Actor: Patient Demographics Consumer

Role: Requests a list of patients matching a minimal set of demographic criteria (e.g., ID or partial name) from the Patient Demographics Supplier. Populates its attributes with demographic information received from the Patient Demographics Supplier.

Actor: Patient Demographics Supplier

Role: Returns demographic information for all patients matching the demographic criteria provided by the Patient Demographics Consumer.

Table 1 lists the transactions for each actor directly involved in the Patient Demographics Query Profile. In order to claim support of this Integration Profile, an implementation must perform the Required transactions (labeled 'R'). Transactions labeled 'O' are Optional. Note that only required transactions defined in the PDQ profile are implemented in this study.

ITI-21 messages are implemented usign HL7[9] Version 2.5. HL7 Version 2.5 standard documents Chapter 3 – Patient Administration (for RSP^K22 message) and Chapter 5 – Query (for QBP^Q22 message) can be investigated for details related to the HL7 Version 2.5 messages.

Table 1. PDQ actors and transactions

Actors	Transactions	Optionality
Patient Demographics Consumer	Patient Demographics Query [ITI-21]	R
	Patient Demographics and Visit Query [ITI-22]	O
Patient Demographics Supplier	Patient Demographics Query [ITI-21]	R
	Patient Demographics and Visit Query [ITI-22]	O

5.4.2 PDQ sequence diagram

Figure 4 depicts sequence diagram of UML showing the order of transactions, that is the messages passed between the actors defined in the PDQ IHE profile. Patient Demographics Query message is realized using HL7 Version 2.5 QBP^Q22 message and Patient Demographics Response message is realized using HL7 Version 2.5 RSP^K22 message.

[9] Recall that HL7 is a framework and related standards for the exchange, integration, sharing and retrieval of electronic health information that supports clinical practice and the management, delivery and evaluation of health services [32].

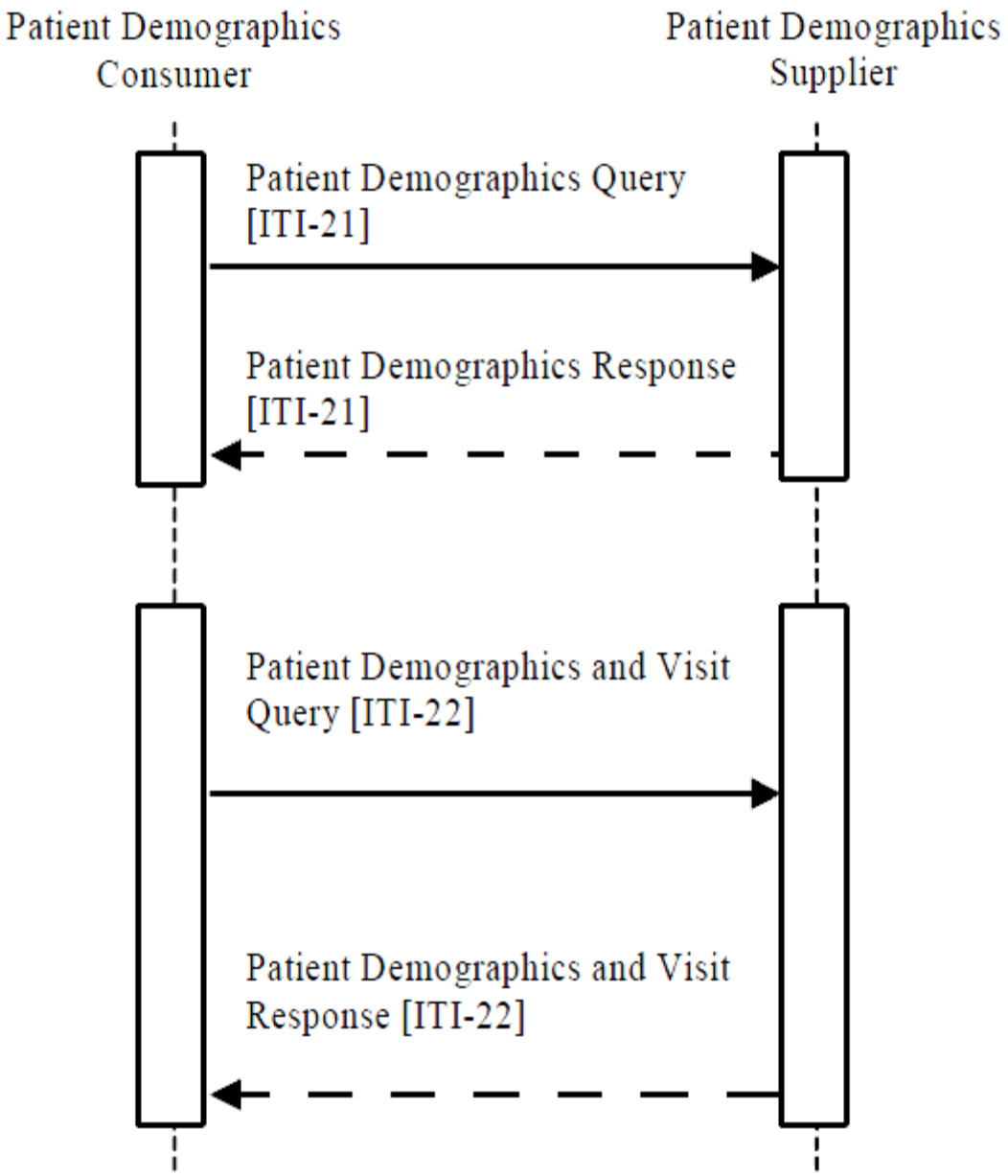

Figure 4. PDQ sequence diagram

5.4.3 PDQ process flow

The Patient Demographics Supplier performs the following functions: It receives patient registration and update messages from other systems in the enterprise (*e.g.* Patient Registration systems). The method in which the Patient Demographics Supplier obtains the updated patient demographic information is not addressed by PDQ profile. So, it is assumed that updated patient demographic information exists in a persistent storage that is accessible by this agent. It is a prerequisite that the

Patient Demographics Supplier possesses current demographic information. One method by which current demographic information may be obtained is for the Patient Demographic Supplier to be grouped with another IHE actor, such as Order Filler, that either maintains or receives such information.

Patient Demographics Supplier responds to queries for information. It receives a Patient Demographics Query or Patient Demographics and Visit Query request from the Patient Demographics Consumer, and returns demographics (and, where appropriate, visit) information from the single domain that is associated with the application to which the query message is sent.

Some use cases are described in the following sections as examples of PDQ profile usage.

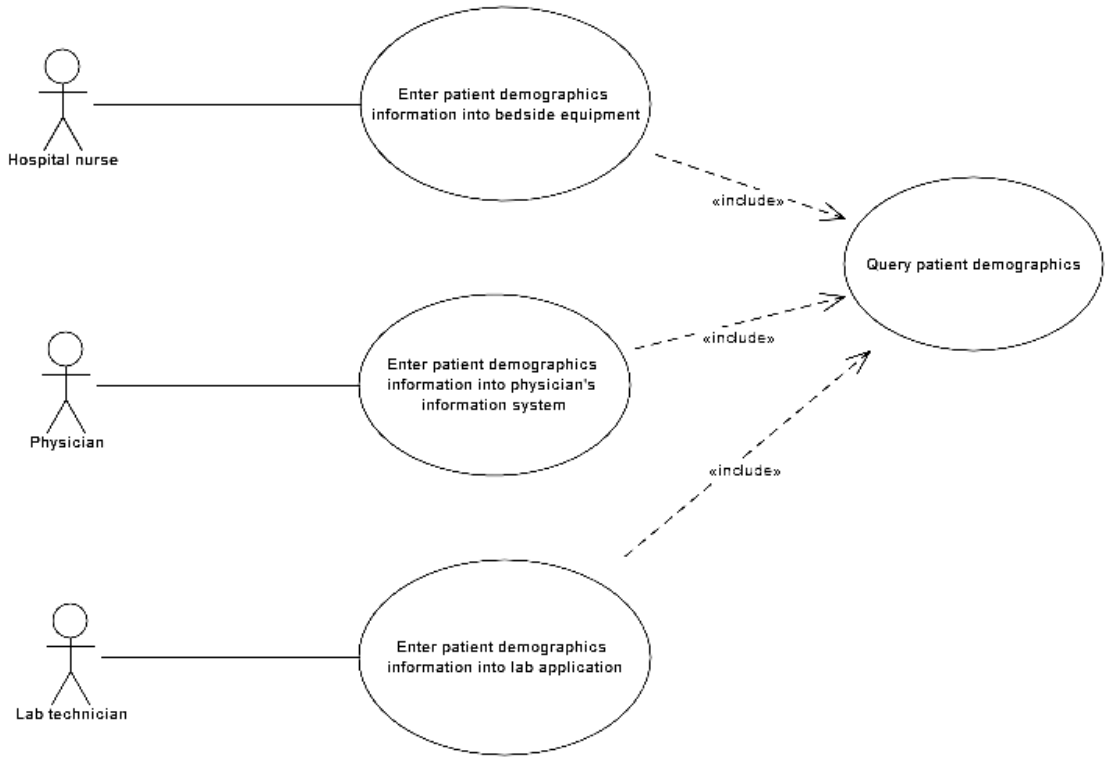

Figure 5. Some clinical use cases which require patient demographics query

Use Case 1: Enter patient information into bedside equipment.

An admitted patient is assigned to a bed. The patient may or may not be able to provide positive ID information. The nurse needs to enter patient identity information into some bedside equipment to establish the relationship of the assigned bed to the patient. The equipment issues a query for a patient pick list to a patient demographics supplier that provides data for a patient pick list. Search criteria entered by the nurse might include one or more of the following:

- Partial or complete patient name (printed on the patient record or told by the patient)
- Patient ID (this may be obtained from printed barcode, a bed-side chart, etc.)
- Date of birth / age range
- Bed ID

The system returns a list of patients showing the full name, age, sex, room/bed, and admit date, and displays the list to the nurse. The nurse then selects the appropriate record to enter the patient identity information into the bedside equipment application.

Use Case 2: Enter patient identity information into physician's information system in his/her office.

A patient visits a physician office for the first time. The physician needs to register the patient, in doing so, it is desired to record the patient's demographic data in the physician's information system. The physician office is connected to a hospital enterprise's central patient registry. The physician issues a patient query request to the central patient registry, with some basic patient demographics data as search criteria. In the returned patient list, the physician picks up an appropriate record for the patient, including the hospital's patient ID, to enter into the system.

The physician's information system may use its own patient identifier, coordinating this identifier with the patient identifier returned in the pick list (sharing the hospital's Patient ID Domain) to retrieve information from the hospital's clinical repository.

Use Case 3: Query patient demographics in an enterprise with multiple patient ID domains.

A lab technician enters some basic demographics data (e.g., patient name) into a lab application to query a patient demographics supplier to identify a patient for his lab exams. As the application also needs the patient identifier in another Patient ID Domain in the enterprise for results delivery, the application is configured to receive patient IDs from other domains in the query response.

Figure 5 depicts UML use case diagrams related mentioned use cases.

5.4.4 Transactions between actors

PDQ defines four transactions passed between the actors Patient Demographics Consumer and Patient Demographics Supplier as shown in Table 1. In this study, transactions Patient Demographics Query and Patient Demographics Reply are implemented.

5.4.4.1 Patient demographics query

Transaction ITI-21 is used by the Patient Demographics Consumer and Patient Demographics Supplier. This transaction involves a request by the Patient Demographics Consumer Actor for information about patients whose demographic data match data provided in the query message. The request is received by the Patient Demographics Supplier Actor. The Patient Demographics Supplier Actor immediately processes the request and returns a response in the form of demographic information for matching patients.

The Patient Demographics Query is conducted by the HL7 QBP^Q22 message. The Patient Demographics Consumer actor shall generate the query message whenever it needs to select from a list of patients whose information matches a minimal set of demographic data.

The segments contained in the HL7 QBP^Q22 message are summarized in Table 2. The information contained in the segments, that is, the related fields and its semantics defined for each segment of the HL7 QBP^Q22 message can be found in

the Appendix A. This information is vital to the implementation, since PDQ profiles mandates certain values to the certain segment fields, and define query semantics.

The receiver shall respond to the query by sending the RSP^K22 message. This satisfies the requirements of acknowledgment; no intermediate ACK message (HL7 Acknowledgement Message) is to be sent.

Table 2. PDQ Patient Demographics Request (HL7 QBP^Q22) segments

QBP	Query by Parameter	Chapter in HL7 2.5
MSH	Message Header	2
QPD	Query Parameter Definition	5
RCP	Response Control Parameter	5
[DSC]	Continuation Pointer	2

Each Patient Demographics Query request specifies two distinct concepts. The Patient Demographics Query is always targeted at a single source of patient demographic information (referred to in this Transaction as the patient information source). A Patient Demographics Supplier may have knowledge of more than one source of demographics. A Patient Demographics Supplier shall support at least one source of patient demographics and may support multiple sources of demographics.

Detailed information of segments contained in PDQ Patient Demographics Request (HL7 QBP^Q22) are given in Appendix A.

5.4.4.2 Patient demographics response

The Patient Demographics Response is conducted by the HL7 RSP^K22 message. The Patient Demographics Supplier Actor shall generate this message in direct response to the QBP^Q22 message previously received. This message satisfies the Application Level, Original Mode Acknowledgement for the HL7 QBP^Q22 message.

The segments of the message listed without enclosing square brackets in the Table 3 below are required. Detailed descriptions of all segments listed in the table below are provided in the following subsections. Other segments of the message are optional.

Table 3. PDQ Patient Demographics Response (HL7 RSP^K22) segments

RSP	Segment Pattern Response	Chapter in HL7 2.5
MSH	Message Header	2
MSA	Message Acknowledgement	2
[{ERR}]	Error	2
QAK	Query Acknowledgement	5
QPD	Query Parameter Definition	5
[{ PID	Patient Identification	3
[PD1]		
[QRI] }]	Query Response Instance	5
[DSC]	Continuation Pointer	2

Detailed information of segments contained in PDQ Patient Demographics Response (HL7 RSP^K22) are given in Appendix B.

Many PDQ query and response examples are provided in Appendix C.

5.5 Implementation Architecture

Logical implementation architecture is summarized in Figure 6. On the server side, Mule ESB is deployed with two units connected. Patient demographics database is connected to the ESB using the database connection mechanism provided by the Mule ESB. This is where the demographics information of the patients is stored. Patient demographics suppier is a web service deployed on the ESB, and managed by the ESB as a service component. On the client side, several patient demographics consumers are shown, which are connect to the ESB acting as a service provider for the patient demographics supplier web service.

Patient demographics consumers are implemented as standalone Java swing applications. They connect to the patient demographics supplier web service, and send a message containing patient demographics query. Patient demographics supplier receives the patient demographics query requests from patient demographics consumers, queries the patient demographics database, and returns the results to the patient demographics consumers as a patient demographics query response message.

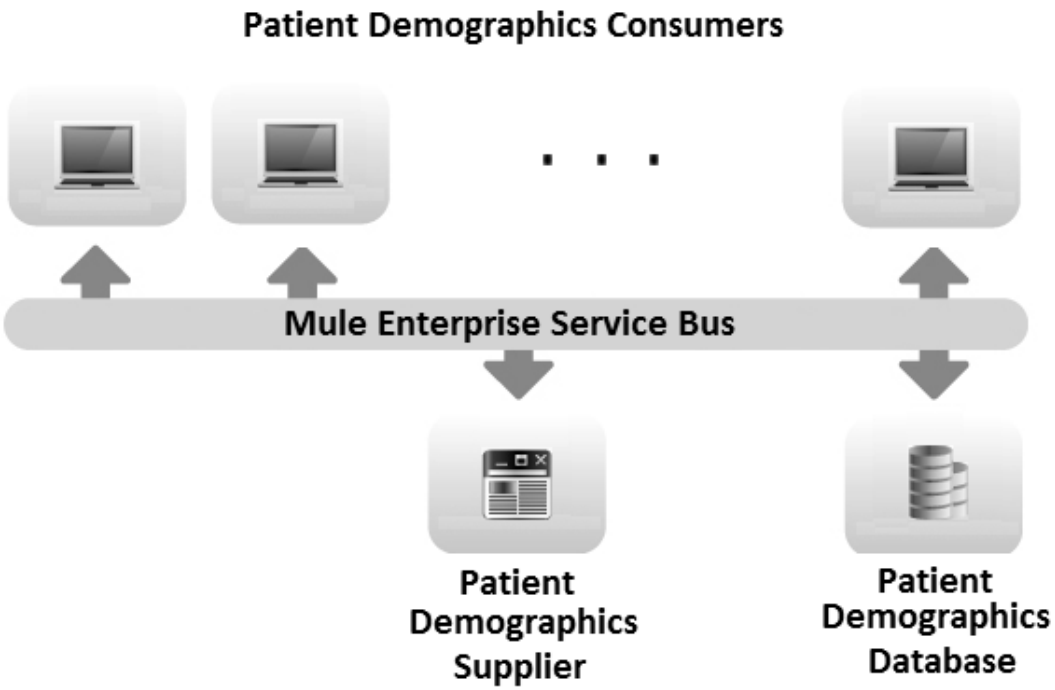

Figure 6. Logical architecture of PDQ implementation

5.6 Technologies Used in PDQ Implementation

- Java programming language: Used for implementing PDQ actors. Java SE 6 Update 24 JDK is used [35].
- Eclipse IDE: Eclipse is used as a development environment for implementing PDQ actors [36].

- HAPI open source HL7 application programming interface. HAPI is used for parsing HL7 messages sent by Patient Demographics Consumers to Patient Demographics Supplier into Java data structures. HAPI is also used for generating HL7 messages sent as a reply by Patient Demographics Supplier to Patient Demographics Consumers [37].

- Apache CXF v2.3.3 open source web services framework. CXF is used to develop PDQ web service that Patient Demographics Supplier provides. Patient Demographics Consumers communicate with PDQ web service. CXF allows developed services can talk different protocols such as SOAP, RESTful HTTP, or XML HTTP. Variety of transports such as JBI, HTTP, or JMS are supported by CXF. PDQ web service implemented as an application in this study supports SOAP over HTTP [38].

- Apache Axis2 v1.5.4 web services framework is used in clients (Patient Demographics Consumers) [39].

- Mule ESB community edition 3.1.1 is used as an ESB, on which PDQ web service is deployed [40].

- Oracle Database 10g is used as a database storing Patient Demographics information [41].

5.7 PDQ Service Implementation Details

A PDQ web service is implemented, which fulfills the IHE PDQ Patient Demographics Supplier role. PDQ service has only one operation, query, with the following signature:

```
public String patientDemographicsQuery(String v25__QBP_Q22)
      throws PDQServiceException
```

The query has only one parameter, v25__QBP_Q22 of data type String. This variable holds HL7 QBP^Q22 message sent by a Patient Demographics Consumer actor, containing information related to a patient whose demographics information is

requested. This string containing HL7 message is parsed by the HAPI, and a QBP object containing all the parameters supplied in the patient demographics query message is constructed to be processed by the service.

The service processes the QBP object and queries the patient demographics database. The result set of the query contains patients to be sent to the Patient Demographics Consumer as the Patient Demographics Query Response message. Note that more than one patient can be located conforming to the fields supplied in the QBP^Q22 message.

The QBP message contains an HL7 QPD (Query Parameter Definition) segment. This segment contains one or more repetitions of the Field *QPD-3-Demographics Fields.* Every instance of QPD-3 field contains two components that together contain the name and value of a distinct demographics parameter to be used in the query. According to the PDQ specification, Patient Demographics Consumer may query and Patient Demographics Supplier should support the fields mentioned in the Table A2 in Appendix B. These are Patient Identifier List (PID.3), Patient Name (PID.5), Date/Time of Birth (PID.7), Administrative Sex (PID.8), Patient Address (PID.11), Patient Account Number (PID.18). For example, the following QPD field given as a query parameter:

@PID.5.1.1^SMITH~@PID.8^F

requests all patients whose PID-5-Patient Name matches the value “SMITH” and whose sex is female. According to the PDQ specification, the Patient Demographics Supplier shall return demographic records that reflect the best fit to all of the supplied search criteria. Obviously there can be more than one patient record corresponding to the given criteria. All the matching patients should be returned in the Patient Demographics Response sent to the Patient Demographics Consumer.

After processing the QBP object, the service constructs a database query (or more than one database queries in case more than one QPD segment is provided in the QBP message), and collects list of matching patient records.

Next, PDQ service instantiates a RSP object using HAPI , then fills the object with the demograpics information belonging to the matching patients (if any) by iterating the matching patient record list.

After the RSP object is constructed, this object is rendered into a String object containing encoded HL7 message by HAPI:

```
ca.uhn.hl7v2.parser.Parser  parser = new
           ca.uhn.hl7v2.parser.PipeParser();
String encodedMessage = parser.encode(rsp);
```

Encoded message is the return value of the service operation patientDemographicsQuery, containing the HL7 RSP^K22 message.

5.8 ESB Implementation

Mule 3 Community Edition is used as an ESB product in the implementation. Mule uses the units called Service Component as building blocks. Mule documentation defines a Service Component as “The key part of the service is the service component. The service component executes business logic on messages, such as reading the invoice object, adding information to it from the customer database, and then forwarding it to the order fulfillment application.” [40].

So, every Service Component receives a message, contains a business logic that is executed with the received message provided as a parameter, and may produce a new message, which then can be sent to another application, or service component.

PDQ service is implemented as a POJO (Plain Old Java Object). Mule contains a mechanism that can be used to publish a POJO as a web service. In order to accomplish this, Mule creates a service component holding the POJO containing the business logic (that is, the operations of the created web service), and exposes this service component as a service provider. The public methods defined in the POJO becomes operations of the published web service. Mule instance can be configured to use various web services frameworks to be used to publish a particular POJO as a web service. In this implementation, Apache CXF framework is selected to be used as a web service provider implementation.

PDQService class is a POJO with only one public method: patientDemographicsQuery() which is exposed as the single operation of the PDQService web service by the ESB. The WSDL of the published PDQService is provided by the ESB.

5.9 Client Implementation

Client of the PDQService web service is implemented using Apache Axis2 web services framework. Axis2 has a tool called wsdl2java, which is a java executable used to generate web service stub and other related java classes given a web service WSDL URI. The gererated classes are sufficient to call operations of the source web service.

In PDQ implementation, the client is Patient Demographics Consumer. Patient Demographics Consumer is responsible for calling the patientDemographicsQuery operation of the PDQService web service and supplying the v25__QBP_Q22 operation parameter holding HL7 QBP^Q22 message.

HAPI is used in the Patient Demographics Consumer client project to create HL7 QBP^Q22 message and parse HL7 RSP_K22 message returned from the PDQService containing the results of the patient demographics query.

6 SUMMARY AND CONCLUSIONS

6.1 Summary

In this study, SOA, the architectural approach commonly used in many contemporary large scale applications, is investigated. SOA allows ICT organizations to create applications by bringing together components of application functionality called services. Services are discrete sets of functionality that are completely separate from each other but can work together.

SOA is an architectural style for building software applications that use services available in a network such as the web. Applications in SOA are built based on services. It promotes loose coupling between services, so that they can be reused. A service is an implementation of a well-defined business functionality and such services can then be consumed by clients in different applications or business processes.

Since each service stands alone, services can be used as building blocks for multiple processes and do not have to be recreated for each type of process or message. This modular approach allows creation of functionality once and makes it possible to re-use it as many times as needed, streamlining development.

Using SOA, businesses can realize dramatic savings on development costs and can rapidly adapt to changing business conditions by reusing and reconfiguring existing services in developing new applications. SOA also enables better integration of enterprise IT resources, including previously isolated application silos and legacy systems.

In the thesis, eight characteristics of services including reusability and the advantages of using SOA especially in the field of application integration are explained in detail.

Web services, the most popular technology to implement SOA systems and two product categories relevant to SOA, namely ESB and BPM are also defined in the thesis and discussed. A typical scenario incorporating web services technology is

given. Also, the advantages and disadvantages of the web services technology are underlined.

Lastly, as an application of SOA, IHE profile PDQ (Patient Demographics Query) is implemented using web services technology and deployed on an ESB acting as a service provider.

6.2 Conclusions

Merging SOA and web services technology enables true enterprise application integration, since web services give SOA applications the ability to easily integrate with applications which are written in a different programming language, running on different operating system and deployed on different hardware platform.

In IHE PDQ implementation completed as an application of SOA in the thesis, PDQ actors are considered as web services. Transactions containing HL7 messages are implemented as SOAP messages. Thinking an application as a collection of services, each containing related operations, greatly simplifies the application design. Once PDQ actors are realized as web services, and the IHE transactions are considered as web service operations, the implementation was straight forward.

In the PDQ implementation, the most problematic part was learning HL7 V2.5 messaging standard. HL7 V2.5 messages do not contain structure information as XML messages do. They are plain text messages, containing different segments each starting at a new line. Every segment has many fields separated by pipe (‘|’) character. Both constructing and parsing messages is difficult, since one has to know which field in the segment contains what information, and the messages themselves are hardly human readable. In order to minimize such a difficulty, an open source HL7 API named HAPI is used. HAPI was really useful, since it greatly reduced development time and effort by providing an API which is able to automatically parse HL7 V2.5 messages into a Java data structure, and construct an HL7 V2.5 message from a given Java HL7 message object containing aggregated segments.

Tools such as ESBs greatly simplfy SOA application development, and brings agility to SOA implementations. ESBs provide many services to SOA applications which eliminate complexities related to SOA application deployment and maintenance.

Another open source software, Mule ESB 3 Community Edition, is used as a service provider providing the web service playing the role of Patient Demographics Supplier PDQ actor. Mule ESB was quite easy to use, but difficult to master. It was easy to perform common tasks such as exposing a java class as a web service, but changing the web services framework used by Mule to CXF framework was quite difficult. This was mainly because lack of Mule documentation on the subject, which is a common problem with some open source tools and libraries.

Mule has an Eclipse plug-in, which can be used in developing mule service containers within the development environment. It has many wizards which help development process and reduce development time. Wizards can perform tasks such as running the mule standalone server and automatically deploying developed classes from within the Eclipse environment.

CXF and AXIS2 are well documented open source Java web services frameworks. They both are mature, and easy to use. Therefore, they are used in server and cliend sides to enable web services technology. For example, AXIS2 wsdl2java tool is used to generate stub class for the PDQ web service hosted by ESB from the WSDL generated by CXF framework. CXF and AXIS2 seamlessly worked together, thanks to web services technology, and Patient Demographics Consumer programs easily made query requests to the ESB. This web service can be called from Patient Demographics Consumers written in different programming languages, since web services are programming language agnostic.

Moreover, since Patient Demographics Supplier is deployed on an ESB, it can use many services provided by Mule ESB product, such as log management, transaction management, transformation support, transportation management, routing support, security, etc.

With the help of standards such as web services, and products such as ESB and BPM, SOA provides a level of flexibility.

6.3 Extension of the Study

PDQ transactions Patient Demographics and Visit Query and Patient Demographics and Visit Reply are not implemented in the PDQ implementation provided with the thesis, because of the time limitations. Patient Demographics Query and Patient Demographics Reply implementations reused, and these two transactions can be implemented as a new operation in the PDQ web service.

An IHE PDQ implementation provided with this thesis can be taken as an example and other IHE profiles belonging to various IHE domains can be implemented, since all IHE profiles are similarly structured, and built on similar paradigms.

A BPM product can be used in case PDQ transactions are required to participate in large scale business processes. Such an implementation could not be completed in the thesis because of time limitations, but the current PDQ web service implementation can be reused, since BPM products, just like ESBs, contain mechanisms for exposing java classes as web services.

REFERENCES

[1] ROTEM, A., SOA Defined, http://www.rgoarchitects.com/Files/SOADefined.pdf, 2007, 10.11.2010.

[2] Service Oriented Methodology, White Paper. NextAxiom, 2003.

[3] Service Oriented Architectures, Adobe, White Paper, 2004.

[4] SCHULTE and NATIS, “Service Oriented" Architectures, Part 1, Gartner, 1996.

[5] NATIS, Service-Oriented Architecture Scenario”, http://www.gartner.com/DisplayDocument?doc_cd=114358 , 2003, 15.11.2010.

[6] ERL, T., A brief history of SOA, http://searchsoa.techtarget.com/expert/KnowledgebaseAnswer/0,289625,sid26_gci1238390_mem1,00.html?ShortReg=1&mboxConv=searchSOA_RegActivate_Submit&, 2006, 09.07.2010.

[7] RODEN, M. and LUBLINSKY, B., Applied SOA: Service Oriented Architecture and Design Strategies, 2006.

[8] Service Oriented Architecture Insights from the Front Line, Survey, Published by Freeform Dynamics, 2006.

[9] FRIEVALD, J., iWay SOA Middleware An Agile Framework for Fast, Flexible, Low Risk Service Deployments, iWay Software, 2006.

[10] From Pilot to Payoff: Service Oriented Architecture Hits it Stride, InfoWorld, White Paper, 2006.

[11] Wikipedia, http://en.wikipedia.org/wiki/Service_(systems_architecture), 2009, 13.07.2010.

[12] ERL, T., The principles of service-orientation part 1 of 6: Introduction to service-orientation, http://searchwebservices.techtarget.com/tip/1,289483,sid26_gci1165286,00.html, 2006, 15.06.2010.

[13] ERL, T., The principles of service-orientation part 2 of 6: Service contracts and loose coupling, http://searchwebservices.techtarget.com/tip/0,289483,sid26_gci1171966,00.html, 2006, 15.06.2010.

[14] ERL, T., The principles of service-orientation part 3 of 6: Service abstraction and reuse, http://searchwebservices.techtarget.com/tip/0,289483,sid26_gci1179915,00.html, 2006, 15.06.2010.

[15] ERL, T., The principles of service-orientation part 4 of 6: Service discoverability and composition, http://searchwebservices.techtarget.com/tip/0,289483,sid26_gci1187364,00.html, 2006, 15.06.2010.

[16] POULIN, M., Evolution of principles of Service Orientation: Service Autonomy, http://www.ebizq.net/blogs/service_oriented/2009/02/evolution_of_principles_of_service_orientation_service_autonomy_part_5.php, 2009, 12.8.2010.

[17] ERL, Thomas, The principles of service-orientation part 5 of 6: Service autonomy and statelessness, http://searchwebservices.techtarget.com/tip/0,289483,sid26_gci1192369,00.html, 2006, 17.06.2010.

[18] New to SOA and Web services, IBM. http://www-128.ibm.com/developerworks/webservices/newto/, 2004, 15.12.2010.

[19] SHAILESH, S., Web Services and SOA History, http://www.saptechies.com/web-services-and-soa-history/, 2005, 24.11.2010.

[20] PATRICK, P, “ Impact of SOA on Enterprise Information Architectures.”, 2005.

[21] MCGOVERN, SAMEER, STEVENS and MATHEWW, “Java Web Services Architecture”, 2006.

[22] http://en.wikipedia.org/wiki/Canonical, 25,02,2011.

[23] ARSANJANI, A, “Toward a pattern language for Service-Oriented Architecture and Integration, Part 1: Build a service eco-system.”, 2005.

[24] SMITH, H., “Business Process Management (BPM): The Third Wave”, 2003.

[25] HARVEY, M., “Essential Business Process Modeling”, 2005.

[26] DOGAC, NAMLI, OKCAN, LALECI, KABAK and EICHELBERG, “Key Issues of Technical Interoperability Solutions in eHealth and the RIDE project”, http://www.google.com/url?sa=t&source=web&cd=5&ved=0CDkQFjAE&url=http%3A%2F%2Fwww.epractice.eu%2Ffiles%2Fmedia%2Fmedia_754.pdf&ei=Oa-2TdngBcrFswaWhdzfDQ&usg=AFQjCNH_Ai6LuedZFRClZzS7YnZI3k9QlA , 2007, 15.04.2011.

[27] Radiological Society of North America (RSNA), http://www.rsna.org/About/index.cfm, 21.02.2011.

[28] Healthcare Information and Management Systems Society (HIMSS), http://www.himss.org/ASP/aboutHimssHome.asp, 21.02.2011.

[29] American College of Cardiology (ACC), http://www.cardiosource.org/ACC/About-ACC.aspx, 21.02.2011.

[30] European Society of Radiology (ESR), http://www.myesr.org/cms/website.php?id=/en/about_esr_ecr.htm, 21.02.2011.

[31] Digital Imaging and Communications in Medicine (DICOM), http://medical.nema.org/dicom/geninfo/Brochure.pdf, 21.02.2011.

[32] Health Level Seven International (HL7), http://www.hl7.org/about/index.cfm?ref=common, 21.02.2011.

[33] IHE Technical Frameworks, http://wiki.ihe.net/index.php?title=Frameworks, 21.02.2011.

[34] IHE Technical Profiles, http://wiki.ihe.net/index.php?title=Profiles, 21.02.2011.

[35] Java Development Kit Downloads, http://www.oracle.com/technetwork/java/javase/downloads/index.html, 02.04.2011

[36] Eclipse IDE (Integrated Development Environment), http://www.eclipse.org

[37] HAPI Open Source Application Programming Interface, http://hl7api.sourceforge.net/, 02.03.2011.

[38] Apache CXF Web Services Framework, http://cxf.apache.org, 21.02.2011.

[39] Apache Axis2 Web Services Framework, http://axis.apache.org/axis2/java/core/, 21.02.2011.

[40] Mule ESB (Enterprise Service Bus), http://www.mulesoft.org, 21.02.2011.

[41] Oracle Database 10g, http://www.oracle.com, 21.02.2011.

APPENDICES

APPENDIX A. HL7 MESSAGE SEGMENT DETAILS OF PATIENT DEMOGRAPHICS QUERY MESSAGE

APPENDIX B. HL7 MESSAGE SEGMENT DETAILS OF PATIENT DEMOGRAPHICS RESPONSE MESSAGE

APPENDIX C. PDQ QUERY AND RESPONSE EXAMPLES

APPENDIX D. CD THAT CONTAINS APPLICATION IMPLEMENTATION

APPENDIX A. HL7 MESSAGE SEGMENT DETAILS OF PATIENT DEMOGRAPHICS QUERY MESSAGE

This section summarizes segment details of HL7 Version 2.5 QBP^Q22 message, that is used as a content for IHE PDQ ITI-21 transaction [Table 2]. The further details can be obtained from IHE web page [33], and HL7 web page [32].

A1. MSH Segment

The Patient Demographics Supplier is able to obtain demographics from at least one and possibly multiple patient information sources. When more than one patient information source is available, Field *MSH-5-Receiving Application* specifies the patient information source that this query is targeting. The Patient Demographics Supplier shall return this value in *MSH-3-Sending Application* of the RSP^K22 response. The value specified in MSH-5 is not related to the value requested in QPD-8 What Domains Returned.

A list shall be published of all Receiving Applications that the Patient Demographics Supplier supports, for the Patient Demographics Consumer to choose from. Each query is processed against one and only one source of patient demographic information.

Field *MSH-9-Message Type* shall have all three components populated with a value. The first component shall have a value of QBP; the second component shall have a value of Q22. The third component it shall have a value of QBP_Q21.

A2. QPD Segment

The Patient Demographics Consumer Actor shall send attributes within the QPD segment as described in Table A1.

The Consumer shall specify “IHE PDQ Query” for QPD-1-Message Query Name. Field *QPD-3-Demographics Fields* consists of one or more repetitions, each of which contains two components that together contain the name and value of a distinct parameter to the query. Acceptable segments are PID and PD1.

Table A1. QPD segment

SEQ	LEN	DT	OPT	TBL#	ITEM#	ELEMENT NAME
1	250	CE	R	0471	01375	Message Query Name
2	32	ST	R+		00696	Query Tag
3		QIP	R			Demographics Fields
8		CX	O			What Domains Returned

The first component of each parameter contains the name of an HL7 element in the form:

@<seg>.<field no>.<component no>.<subcomponent no>

The above format is populated according to common HL7 usage for specifying elements used in query parameters, as follows:

- <seg> represents a 3-character segment ID from the HL7 Standard.
- <field no> is the number of a field within the segment as shown in the SEQ column of the segment attribute table for the segment selected.
- <component no>, for fields whose data types contain multiple components, shall contain the cardinal number of the component being valued. For fields whose data types do not contain multiple components, <component no> shall not be valued and its preceding period shall not appear.
- <subcomponent no>, for components whose data types contain multiple subcomponents, shall contain the cardinal number of the subcomponent being

valued. For components whose data types do not contain multiple subcomponents, <subcomponent no> shall not be valued and its preceding period shall not appear.

The second subcomponent of each parameter contains the value that is to be matched. If it is desired to constrain the quality of a match within the bounds of an algorithm known to the Supplier, the algorithm and constraint values may be specified in Fields QPD-4 through QPD-7.

The Patient Demographics Consumer may specify, and the Patient Demographics Supplier shall support, the fields in the Table A2.

Table A2. QPD-3 fields required to be supported

FLD	ELEMENT NAME
PID.3	Patient Identifier List
PID.5	Patient Name
PID.7	Date/Time of Birth
PID.8	Administrative Sex
PID.11	Patient Address
PID.18	Patient Account Number

The Patient Demographics Supplier shall return demographic records that reflect the best fit to all of the search criteria.

An example of parameter expressions in QPD-3:

@PID.5.1.1^SMITH~@PID.8^F

requests all patients whose family name (first subcomponent (data type ST) of the first component (data type FN) of PID-5-Patient Name (data type XPN)) matches the value “SMITH‟ and whose sex (PID-8-Sex 3625 (data type IS)) matches the value “female‟.

Field QPD-8 restricts the set of domains for which identifiers are returned in PID-3:

1. In a multiple-domain environment, QPD-8 may be used to identify one or more domains of interest to the Patient Demographics Consumer and from which the Consumer wishes to obtain a value for *PID-3-Patient Identifier*. Note that the patient information source designated by MSH-5 may or may not be associated with any of the Patient ID Domains listed in *QPD-8-What Domains Returned*.

If QPD-8 is empty, the Patient Demographics Supplier shall return all Patient IDs known by the Patient Demographics Supplier for each patient that matches the search criteria.

If QPD-8 is specified and the domains are recognized, the Patient Demographics Supplier shall return the Patient IDs for each patient that matches the search criteria.

Any domain not recognized by the Patient Demographics Supplier is an error condition.

2. In a single-domain environment, QPD-8 may be ignored by the Patient Demographics Supplier. The Supplier shall always return the identifier from the Patient ID Domain known by the Patient Demographics Supplier.

Within field QPD-8, only component 4 (Assigning Authority) shall be valued. The Patient Demographics Supplier may or may not be able to supply additional identifiers from the domains specified in QPD-8.

The Patient Demographics Consumer shall be able to support at least one of the following mechanisms for specifying QPD-8:

1. Transmit an empty value and receive all identifiers in all domains known by the Patient Demographics Supplier (one or more domains), or

2. Transmit a single value and receive zero or more identifiers in a single domain, or

3. Transmit multiple values and receive mutiple identifiers in those multiple domains.

A3. RCP Segment

The Patient Demographics Consumer Actor shall send attributes within the RCP segment as described in Table A3. Fields not listed are optional and may be ignored.

Table A3. RCP segment

SEQ	LEN	DT	OPT	TBL#	ITEM#	ELEMENT NAME
1	1	ID	R	0091	00027	Query Priority
2	10	CQ	O	0126	00031	Quantity Limited Request

Field *RCP-1-Query Priority* shall always contain "**I"**, signifying that the response to the query is to be returned in Immediate mode.

The Patient Demographics Consumer Actor may request that responses to the query be sent, using the HL7 Continuation Protocol, in increments of a specified number of patient records. (In the context of the HL7 query, a patient record is defined as the PID segment and any segments accompanying it for each patient.) It is desirable to request an incremental response if the query could result in hundreds or thousands of matches. So, the Patient Demographics Supplier Actor shall support the HL7 Continuation Protocol.

Field RCP-2 is of data type CQ, which contains two components. The first component contains the number of increments, always expressed as an integer greater than 0, while the second component contains the kind of increment, always RD to signify that incremental replies are specified in terms of records.

For example, 50^RD requests 50 records at a time.

A4. DSC Segment

The Patient Demographics Consumer Actor may request additional increments of data by specifying this segment on the query request. This segment should be omitted on the initial query request. Its purpose is to request additional increments of

the data from the Patient Demographic Supplier Actor. Table A4 summarizes fields contained in this segment.

Table A4. DSC segment

SEQ	LEN	DT	OPT	TBL#	ITEM #	ELEMENT NAME
1	180	ST	O		00014	Continuation Pointer
2	1	ID	O	0398	01354	Continuation Style

To request additional increments of data, DSC-1 (Continuation Pointer) shall echo the value from RSP^K22 DSC-1.

DSC-2 (Continuation Style) shall always contain “I”, signifying that this is part of an interactive continuation message.

APPENDIX B. HL7 MESSAGE SEGMENT DETAILS OF PATIENT DEMOGRAPHICS RESPONSE MESSAGE

This section summarizes segment details of HL7 Version 2.5 RSP^K22 message, that is used as a content for IHE PDQ ITI-21 transaction [Table 3]. The further details can be obtained from IHE web page [33], and HL7 web page [32].

B1. MSH Segment

Field *MSH-3-Sending Application* specifies the patient information source that processed the query. The Patient Demographics Supplier shall use Field *MSH-3-Sending Application* of the RSP^K22 message to return the value it received from the Patient Demographics Consumer in Field *MSH-5-Receiving Application* of the QBP^Q22 message.

Field *MSH-9-Message Type* shall have all three components populated with a value. The first component shall have a value of RSP; the second component shall have a value of K22. The third component shall have a value of RSP_K22.

B2. MSA Segment

The Patient Demographics Supplier Actor is not required to send any attributes within the MSA segment beyond what is specified in the HL7 standard.

B3. QAK Segment

The Patient Demographics Supplier Actor shall send attributes within the QAK segment as defined in Table B1.

QAK-1 (Query Tag) shall echo the same value of QPD-2 (Query Tag) of the QBP^Q22 message, to allow the Patient Demographics Query Consumer to match the response to the corresponding query request.

Table B1. QAK segment

SEQ	LEN	DT	OPT	TBL#	ITEM#	ELEMENT NAME
1	32	ST	R		00696	Query Tag
2	2	ID	R+	0208	00708	Query Response Status

B4. QPD Segment

The Patient Demographics Supplier Actor shall echo the QPD Segment value that was sent in the QBP^Q22 message.

B5. PID Segment

The Patient Demographics Supplier Actor shall return one PID segment group for each matching patient record found. The Supplier shall return the attributes within the PID segment as specified in Table B2. In addition, the Patient Demographics Supplier Actor shall return all other attributes within the PID segment for which it is able to supply values.

Table B2. PID segment

SEQ	LEN	DT	OPT	TBL#	ITEM#	ELEMENT NAME
3	250	CX	R		00106	Patient Identifier List
5	250	XPN	R		00108	Patient Name
7	26	TS	R2		00110	Date/Time of Birth
8	1	IS	R2	0001	00111	Administrative Sex
11	250	XAD	R2		00114	Patient Address
18	250	CX	R2		00121	Patient Account Number

The Patient Demographics Supplier may or may not be able to supply additional identifiers from the domains specified in QPD-8. Inability to supply an identifier in a particular domain is not an error, provided that the domain is recognized.

The PID segment and its associated PD1 and QRI segments are returned only when the Patient Demographics Supplier Actor is able to associate the search information in QPD-3 with one or more patient records in the patient information source associated with *MSH-5-Receiving Application*.

B6. QRI Segment

For each patient for which the Patient Demographics Supplier Actor returns a PID Segment, it may optionally return the QRI (Query Response Instance) segment, but is not required to do so.

B7. DSC Segment

If the number of records is specified in *RCP-2-Quantity Limited Request*, the Patient Demographics Supplier Actor shall return an incremental response of that number of records when the number of matching records it finds exceeds the number of records specified in RCP-2.

As long as the Patient Demographics Supplier Actor has records to return in addition to those returned in the incremental response, the Supplier shall return a DSC Segment. The single field of the DSC Segment shall contain a unique alphanumeric value (the Continuation Pointer) that the Patient Demographics Consumer may return in the DSC of the QBP^Q22 message to request the next increment of responses. The Supplier shall return increments as many times as the Consumer requests them (and there are increments to return), and shall stop when the Consumer sends a cancel query (QCN^J01) message (or when there are no more increments to return).

APPENDIX C. PDQ QUERY AND RESPONSE EXAMPLES

Inbound query:

MSH|^~\&|OTHER_IBM_BRIDGE_TLS|IBM|PAT_IDENTITY_X_REF_MGR_MISYS|ALLSCRIPTS|20090226131520060O||QBP^Q22^QBP_Q21|7965847682428535543|P|2.5 QPD|Q22^Find Candidates^HL7|48709646603885995650965675121 28|@PID.5.1.1^MOO* RCP|I|10^RD

Outbound response:

MSH|^~\&|PAT_IDENTITY_X_REF_MGR_MISYS_TLS|ALLSCRIPTS|OTHER_IBM_BRIDGE_TLS|IBM|20090226141518-0500||RSP^K22|OpenPIXPDQ10.243.0.65.19770811493153|P|2.5
MSA|AA|7965847682428535543
QAK|48709646603885995650965675121 28|OK||6|6|0 QPD|Q22^Find Candidates^HL7|48709646603885995650965675121 28|@PID.5.1.1^MOO*
PID|1||LPDQ113XX04^^^IHELOCAL&1.3.6.1.4.1.21367.2009.1.2.310&ISO^PI~PDQ113XX04^^^IHENA&1.3.6.1.4.1.21367.2009.1.2.300&ISO^PI~TEMP000025^^^EMCO&1.3.6.1.4.1.21367.2009.1.2.348&ISO^PI||MOODY^WARREN||19780820|M|||1000 CLAYTON RD^^CLAYTON^MO^63105
PID|2||LPDQ113XX05^^^IHELOCAL&1.3.6.1.4.1.21367.2009.1.2.310&ISO^PI~PDQ113XX05^^^IHENA&1.3.6.1.4.1.21367.2009.1.2.300&ISO^PI~TEMP000026^^^EMCO&1.3.6.1.4.1.21367.2009.1.2.348&ISO^PI~TEMP000026^^^EMCO&1.3.6.1.4.1.21367.2009.1.2.348&ISO^PI||MOONEY^STAN||19780920|M|||100 TAYLOR^^ST LOUIS^MO^63110
PID|3||TEMP000020^^^EMCO&1.3.6.1.4.1.21367.2009.1.2.348&ISO^PI||MOORE^MARK||19380224|F|||^^Chicago^IL^65932
PID|4||TEMP000019^^^EMCO&1.3.6.1.4.1.21367.2009.1.2.348&ISO^PI||MOORE^MARTHA||19820904|F|||^^Gainesville^FL^32609
PID|5||LPDQ113XX02^^^IHELOCAL&1.3.6.1.4.1.21367.2009.1.2.310&ISO^PI~PDQ113XX02^^^IHENA&1.3.6.1.4.1.21367.2009.1.2.300&ISO^PI~PDQ113XX02^^^IHENA&1.3.6.1.4.1.21367.2009.1.2.300&ISO^PI||MOORE^RALPH||19510707|M|||510 S KINGSHIGHWAY^^ST. LOUIS^MO^63110^USA

PID|6||39^^^MIEH&1.3.6.1.4.1.21367.2009.1.2.380&ISO^PI~39^^^MIEH&1.3.6.1.4.1.21367.2009.1.2.380&ISO^PI~LPDQ113XX01^^^IHELOCAL&1.3.6.1.4.1.21367.2009.1.2.310&ISO^PI~PDQ113XX01^^^IHENA&1.3.6.1.4.1.21367.2009.1.2.300&ISO^PI||MOORE^CHIP||19380224|M|||^^^IN

Inbound query:

MSH|^~\&|OTHER_IBM_BRIDGE_TLS|IBM|PAT_IDENTITY_X_REF_MGR_MISYS|ALLSCRIPTS|20090226131524-0600||QBP^Q22^QBP_Q21|438460523393200678 5|P|2.5
QPD|Q22^FindCandidates^HL7|284626628416548310904573937 1027|@PID.3.1^PDQ113XX05~@PID.3.4.1^IHENA~@PID.3.4.2^1.3.6.1.4.1.21367.2009.1.2.300~@PID.3.4.3^ISO RCP|I|10^RD

Outbound response:

MSH|^~\&|PAT_IDENTITY_X_REF_MGR_MISYS_TLS|ALLSCRIPTS|OTHER_IBM_BRIDGE_TLS|IBM|20090226141522-0500||RSP^K22|OpenPIXPDQ10.243.0.65.19770811557491|P|2.5
MSA|AA|4384605233932006785
QAK|284626628416548310904573937 1027|OK||1|1|0
QPD|Q22^FindCandidates^HL7|284626628416548310904573937 1027|@PID.3.1^PDQ113XX05~@PID.3.4.1^IHENA~@PID.3.4.2^1.3.6.1.4.1.21367.2009.1.2.300~@PID.3.4.3^ISO
PID|1||LPDQ113XX05^^^IHELOCAL&1.3.6.1.4.1.21367.2009.1.2.310&ISO^PI~PDQ113XX05^^^IHENA&1.3.6.1.4.1.21367.2009.1.2.300&ISO^PI~TEMP000026^^^EMCO&1.3.6.1.4.1.21367.2009.1.2.348&ISO^PI~TEMP000026^^^EMCO&1.3.6.1.4.1.21367.2009.1.2.348&ISO^PI||MOONEY^STAN||19780920|M|||100 TAYLOR^^ST LOUIS^MO^63110

Inbound query:

MSH|^~\&|OTHER_IBM_BRIDGE_TLS|IBM|PAT_IDENTITY_X_REF_MGR_MISYS|ALLSCRIPTS|20090226131528-0600||QBP^Q22^QBP_Q21|661520573601174 3610|P|2.5
QPD|Q22^FindCandidates^HL7|531922740437920845303017257 0701|@PID.3.1^PDQ113*~@PID.3.4.1^IHENA~@PID.3.4.2^1.3.6.1.4.1.21367.2009.1.2.300~@PID.3.4.3^ISO RCP|I|10^RD

Outbound response:

MSH|^~\&|PAT_IDENTITY_X_REF_MGR_MISYS_TLS|ALLSCRIPTS|OTHER_IBM_BRIDGE_TLS|IBM|20090226141526-0500||RSP^K22|OpenPIXPDQ10.243.0.65.19770811618339|P|2.5
MSA|AA|6615205736011743610
QAK|531922740437920845303017257070l|OK||5|5|0
QPD|Q22^FindCandidates^HL7|531922740437920845303017257070l|@PID.3.1^PDQ113*~@PID.3.4.1^IHENA~@PID.3.4.2^1.3.6.1.4.1.21367.2009.1.2.300~@PID.3.4.3^ISO
PID|1||LPDQ113XX03^^^IHELOCAL&1.3.6.1.4.1.21367.2009.1.2.310&ISO^PI~PDQ113XX03^^^IHENA&1.3.6.1.4.1.21367.2009.1.2.300&ISO^PI~TEMP000024^^^EMCO&1.3.6.1.4.1.21367.2009.1.2.348&ISO^PI||MOHR^ALICE||19580131|F|||820 JORIE BLVD.^^OAK BROOK^IL^60523
PID|2||LPDQ113XX04^^^IHELOCAL&1.3.6.1.4.1.21367.2009.1.2.310&ISO^PI~PDQ113XX04^^^IHENA&1.3.6.1.4.1.21367.2009.1.2.300&ISO^PI~TEMP000025^^^EMCO&1.3.6.1.4.1.21367.2009.1.2.348&ISO^PI||MOODY^WARREN||19780820|M|||1000 CLAYTON RD^^CLAYTON^MO^63105
PID|3||LPDQ113XX05^^^IHELOCAL&1.3.6.1.4.1.21367.2009.1.2.310&ISO^PI~PDQ113XX05^^^IHENA&1.3.6.1.4.1.21367.2009.1.2.300&ISO^PI~TEMP000026^^^EMCO&1.3.6.1.4.1.21367.2009.1.2.348&ISO^PI~TEMP000026^^^EMCO&1.3.6.1.4.1.21367.2009.1.2.348&ISO^PI||MOONEY^STAN||19780920|M|||100 TAYLOR^^ST LOUIS^MO^63110
PID|4||LPDQ113XX02^^^IHELOCAL&1.3.6.1.4.1.21367.2009.1.2.310&ISO^PI~PDQ113XX02^^^IHENA&1.3.6.1.4.1.21367.2009.1.2.300&ISO^PI~PDQ113XX02^^^IHENA&1.3.6.1.4.1.21367.2009.1.2.300&ISO^PI||MOORE^RALPH||19510707|M|||510 S KINGSHIGHWAY^^ST. LOUIS^MO^63110^USA
PID|5||39^^^MIEH&1.3.6.1.4.1.21367.2009.1.2.380&ISO^PI~39^^^MIEH&1.3.6.1.4.1.21367.2009.1.2.380&ISO^PI~LPDQ113XX01^^^IHELOCAL&1.3.6.1.4.1.21367.2009.1.2.310&ISO^PI~PDQ113XX01^^^IHENA&1.3.6.1.4.1.21367.2009.1.2.300&ISO^PI||MOORE^CHIP||19380224|M|||^^^IN

Inbound query:

MSH|^~\&|OTHER_IBM_BRIDGE_TLS|IBM|PAT_IDENTITY_X_REF_MGR_MISYS|ALLSCRIPTS|20090226131532-0600||QBP^Q22^QBP_Q21|8635581430489875581|P|2.5 QPD|Q22^Find Candidates^HL7|664470896599749451216175886424 4|@PID.7.1^19780920 RCP|I|10^RD

Outbound response:

MSH|^~\&|PAT_IDENTITY_X_REF_MGR_MISYS_TLS|ALLSCRIPTS|OTHER_IBM_BRIDGE_TLS|IBM|20090226141529-0500||RSP^K22|OpenPIXPDQ10.243.0.65.19770811677697|P|2.5 MSA|AA|8635581430489875581 QAK|664470896599749451216175886424 4|OK||2|2|0 QPD|Q22^Find Candidates^HL7|664470896599749451216175886424 4|@PID.7.1^19780920 PID|1||LPDQ113XX05^^^IHELOCAL&1.3.6.1.4.1.21367.2009.1.2.310&ISO^PI~PDQ113XX05^^^IHENA&1.3.6.1.4.1.21367.2009.1.2.300&ISO^PI~TEMP000026^^^EMCO&1.3.6.1.4.1.21367.2009.1.2.348&ISO^PI~TEMP000026^^^EMCO&1.3.6.1.4.1.21367.2009.1.2.348&ISO^PI||MOONEY^STAN||19780920|M|||100 TAYLOR^^ST LOUIS^MO^63110 PID|2||TEMP000018^^^EMCO&1.3.6.1.4.1.21367.2009.1.2.348&ISO^PI||RICHTER^CHARLES||19780920|M|||^^Gainesville^FL^32609

Inbound query:

MSH|^~\&|OTHER_IBM_BRIDGE_TLS|IBM|PAT_IDENTITY_X_REF_MGR_MISYS|ALLSCRIPTS|20090226131536-0600||QBP^Q22^QBP_Q21|7597683905236566560|P|2.5 QPD|Q22^Find Candidates^HL7|84947641013036616688906458335 27|@PID.5.1.1^MO*~@PID.8^F RCP|I|10^RD

Outbound response:

MSH|^~\&|PAT_IDENTITY_X_REF_MGR_MISYS_TLS|ALLSCRIPTS|OTHER_IBM_BRIDGE_TLS|IBM|20090226141533-0500||RSP^K22|OpenPIXPDQ10.243.0.65.19770811737041|P|2.5 MSA|AA|7597683905236566560 QAK|84947641013036616688906458335 27|OK||3|3|0 QPD|Q22^Find

Candidates^HL7|84947641013036616688906458335 27|@PID.5.1.1^MO*~@PID.8^F

PID|1||LPDQ113XX03^^^IHELOCAL&1.3.6.1.4.1.21367.2009.1.2.310&ISO^PI~PDQ113XX03^^^IHENA&1.3.6.1.4.1.21367.2009.1.2.300&ISO^PI~TEMP000024^^^EMCO&1.3.6.1.4.1.21367.2009.1.2.348&ISO^PI||MOHR^ALICE||19580131|F|||820 JORIE BLVD.^^OAK BROOK^IL^60523

PID|2||TEMP000020^^^EMCO&1.3.6.1.4.1.21367.2009.1.2.348&ISO^PI||MOORE^MARK||19380224|F|||^^Chicago^IL^65932

PID|3||TEMP000019^^^EMCO&1.3.6.1.4.1.21367.2009.1.2.348&ISO^PI||MOORE^MARTHA||19820904|F|||^^Gainesville^FL^32609

Inbound query:

MSH|^~\&|OTHER_IBM_BRIDGE_TLS|IBM|PAT_IDENTITY_X_REF_MGR_MISYS|ALLSCRIPTS|20090226131539-0600||QBP^Q22^QBP_Q21|467926674483766 8258|P|2.5 QPD|Q22^Find Candidates^HL7|131427392849992352517538089 2595|@PID.5.1.1^MOORE~@PID.7.1^19380224 RCP|I|10^RD

Outbound response:

MSH|^~\&|PAT_IDENTITY_X_REF_MGR_MISYS_TLS|ALLSCRIPTS|OTHER_IBM_BRIDGE_TLS|IBM|20090226141537-0500||RSP^K22|OpenPIXPDQ10.243.0.65.19770811796403|P|2.5

MSA|AA|4679266744837668258

QAK|131427392849992352517538089 2595|OK||2|2|0 QPD|Q22^Find Candidates^HL7|131427392849992352517538089 2595|@PID.5.1.1^MOORE~@PID.7.1^19380224

PID|1||TEMP000020^^^EMCO&1.3.6.1.4.1.21367.2009.1.2.348&ISO^PI||MOORE^MARK||19380224|F|||^^Chicago^IL^65932

PID|2||39^^^MIEH&1.3.6.1.4.1.21367.2009.1.2.380&ISO^PI~39^^^MIEH&1.3.6.1.4.1.21367.2009.1.2.380&ISO^PI~LPDQ113XX01^^^IHELOCAL&1.3.6.1.4.1.21367.2009.1.2.310&ISO^PI~PDQ113XX01^^^IHENA&1.3.6.1.4.1.21367.2009.1.2.300&ISO^PI||MOORE^CHIP||19380224|M|||^^^IN

Inbound query:

MSH|^~\&|OTHER_IBM_BRIDGE_TLS|IBM|PAT_IDENTITY_X_REF_MGR_MISYS|ALLSCRIPTS|20090226131543-0600||QBP^Q22^QBP_Q21|68808813780998744844|P|2.5
QPD|Q22^FindCandidates^HL7|605865177510461743892216624 2613|@PID.5.1.1^MOORE~@PID.11.1.1^10 PINETREE RCP|I|10^RD

Outbound response:

MSH|^~\&|PAT_IDENTITY_X_REF_MGR_MISYS_TLS|ALLSCRIPTS|OTHER_IBM_BRIDGE_TLS|IBM|20090226141540-0500||RSP^K22|OpenPIXPDQ10.243.0.65.19770811854243|P|2.5
MSA|AA|6880881378099874844
QAK|605865177510461743892216624 2613|OK||1|1|0
QPD|Q22^FindCandidates^HL7|605865177510461743892216624 2613|@PID.5.1.1^MOORE~@PID.11.1.1^10 PINETREE
PID|1||TEMP000020^^^EMCO&1.3.6.1.4.1.21367.2009.1.2.348&ISO^PI||MOORE^MARK||19380224|F|||10 PINETREE^^Chicago^IL^65932

Inbound query:

MSH|^~\&|OTHER_IBM_BRIDGE_TLS|IBM|PAT_IDENTITY_X_REF_MGR_MISYS|ALLSCRIPTS|20090226131547-0600||QBP^Q22^QBP_Q21|28725422760091941 18|P|2.5
QPD|Q22^FindCandidates^HL7|76866848180863854926834452205 29|@PID.5.1.1^OTHER_IBM_BRIDGE RCP|I|10^RD

Outbound response:

MSH|^~\&|PAT_IDENTITY_X_REF_MGR_MISYS_TLS|ALLSCRIPTS|OTHER_IBM_BRIDGE_TLS|IBM|20090226141544-0500||RSP^K22|OpenPIXPDQ10.243.0.65.19770811913601|P|2.5
MSA|AA|2872542276009194118
QAK|76866848180863854926834452205 29|OK||2|2|0 QPD|Q22^Find Candidates^HL7|76866848180863854926834452205 29|@PID.5.1.1^OTHER_IBM_BRIDGE
PID|1||103^^^IBOT&1.3.6.1.4.1.21367.2009.1.2.370&ISO^PI||OTHER_IBM_BRIDGE^MARION||19661109|F

PID|2||301^^^IBOT&1.3.6.1.4.1.21367.2009.1.2.370&ISO^PI||OTHER_IBM_BRIDGE
^SONDRA||19671109|F