PPEPR: Plug and Play Electronic Patient Records

Ratnesh Sahay
DERI
National University of Ireland, Galway
ratnesh.sahay@deri.org

Waseem Akhtar
DERI
National University of Ireland, Galway
waseem.akhtar@deri.org

Ronan Fox
DERI
National University of Ireland, Galway
ronan.fox@deri.org

ABSTRACT

The integration of Electronic Patient Record (EPR) systems is at the centre of many of the new regional and national initiatives to integrate clinical processes across department, region, and national levels. Web Service technologies offer significant solutions to provide an interoperable communication infrastructure but are unable to support precise definitions for healthcare messages, functionality, and standards, required for making meaningful integration. The lack of interoperability within healthcare standards adds complexity to the initiatives. This heterogeneity exists within two versions of same standard (e.g. HL7), and also between standards (e.g. HL7, openEHR, CEN TC/251 13606).

We therefore introduce an integration platform PPEPR (Plug and Play Electronic Patient Records), which is based on the principles of a semantic Service-Oriented Architecture (sSOA). PPEPR solves the problem of interoperability at the semantic level. A key focus of PPEPR is that once a patient information is captured, should be available for use across all potential care processes.

Categories and Subject Descriptors

D.2.12 [**Software**]: Software Architectures;
Interoperability[Domain-specific architectures]

Keywords

eHealth, SOA, Web services, Semantic Interoperability

1. INTRODUCTION

Healthcare enterprise applications (e.g. Electronic Patient Record Systems (EPRs)[1])have been created within hospitals to address specific end user requirements. The EPRs, however, exist as islands of information with little or no connectivity between them. What connectivity has been implemented has been a largely manual effort with significant resources spent in routing and mapping healthcare messages between those systems [8]. In some cases special interfaces have had to be developed to allow EPR systems to communicate. Consider the following set of problems:

- Most hospitals still have obsolete standards or protocols running as their critical applications.
- Healthcare standards are under constant development and improvements.
- IT or healthcare professionals may diverge from the use of healthcare standards' (e.g. HL7[2] v2.x, v3), CEN TC/251 13606[3], openEHR[4], etc.) intended meanings, thus defeating the purpose of those standards.
- Healthcare standards formatted in XML solve the integration or interoperability problem at a syntactic level, but domain specific solutions are required to achieve meaningful integration.

In this paper we introduce PPEPR[5], an integration platform for heterogeneous EPRs. PPEPR is based on the design principles of a semantic SOA Reference Architecture[6] and built (It's first prototype) around semantic Web service technology. The structure of this paper is as follows: first we describe the relation between healthcare, SOA, Web services, and the emergent need of semantics. Then we introduce the PPEPR approach and a simplified version of the PPEPR architecture. Next, we introduce our approach for ontology and an adapter framework development. Finally we explain how PPEPR addresses the EPRs integration requirements.

2. EPRS INTEGRATION: SOA, WEB SERVICES, & SEMANTICS

Gradually, organisations are adopting SOA as their fundamental architecture for systems development[7]. For healthcare systems there are two conceptual viewpoints (both are valid):

- Implementing a general SOA framework (common infrastructure, tools and approaches), "HL7 is just another content type"

[1]http://xml.coverpages.org/hl7PRA.html

[2]http://www.hl7.org
[3]http://www.cen.eu/
[4]http://openehr.org/
[5]http://ppepr.deri.ie/
[6]http://www.oasis-open.org/apps/org/workgroup/semantic-ex/

SAC'08 March 16-20, 2008, Fortaleza, Ceará, Brazil

- Implementing an HL7 based messaging architecture that can use different messaging and transports, including Web services.

The first tends to lead to the conclusion that HL7 should just define content and the second suggests HL7 should define the whole stack. Various initiatives focus on each of the viewpoints; creating an overall enterprise SOA where HL7 is just one content type amongst others (e.g. X.12[7]), and the need for HL7 to fully define an operating model for those using less complete frameworks such as basic HL7 LLP[8] messaging. These divergent viewpoints map to issues on the use of the various wrapper layers around HL7 content, the level of overlap between HL7 functionality, and advanced general frameworks such as Web Services. The majority of existing EPRs do not employ Web services, creating significant challenges for any integration system based on advanced research/technology (e.g. semantic Web services), without imposing constraints on existing systems (e.g. requiring the upgrade of existing EPRs to use Web services). This transition to the use of Web services imposes significant costs to the implementing enterprise. To counter this difficulty PPEPR provides three types of integration between non-Web services and Web services enabled EPRs.

Web services provide the technology foundation for implementing and delivering SOA platforms. However, SOA itself is not a complete solution for the integration of healthcare information systems since the two core challenges of conventional computing - search and integration - (also known as semantic gap of SOA) are not addressed. Therefore, SOA significantly and fundamentally depends on solutions to fill the semantic gap so that its full potential can be achieved[1, 2]. The PPEPR architecture employs a semantic Service-Oriented Architecture (sSOA) and thus acts as such a solution. The architecture will be realized by developing a cross functional, semantically enabled integration platform with input from Galway University Hospital (GUH)[9].

3. PPEPR APPROACH

The development approach is divided into two steps: First, Modelling the healthcare domain knowledge at a conceptual level (during design time) by, (a) annotating healthcare functionalities (Functional ontology) (b) annotating healthcare messages (Message ontology) and, (c) mapping ontologies at the schema level. Second, Integrate EPRs (during run time) using the modelled concepts by, (a) mediating ontologies at instance level (b) mediating healthcare messages and functionalities at instance level. The above approach helps to delineate scope and responsibilities as well as organise, maintain and access the healthcare data, information and knowledge. The key outcomes of PPEPR are Ontologies (Message, Functional) and the Adapter Framework to connect to traditional healthcare software.

4. PPEPR ARCHITECTURE

Figure 1 shows the simplified architecture of PPEPR. The main components are:

EPRs: In the first prototype, PPEPR incorporates HL7 (2.x, v3, and CDA) compliant EPRs . In advanced versions of PPEPR, openEHR and CEN TC/251 13606 will be incorporated within the integration framework.

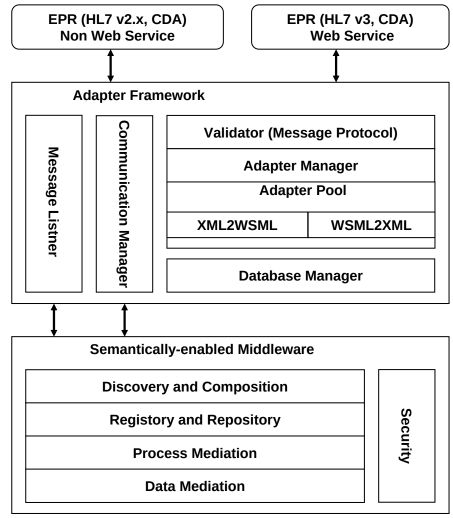

Figure 1: Simplified PPEPR architecture

Adapter Framework: The adapter framework is a common platform for enabling communication between systems that use different data formats and it is an application specific software component that transforms data from its source to target format. The framework provides an adapter skeleton and allows developers to register (plug-in) their adapters and invokes the target application through one of the registered adapters. Direct invocation is also possible if the input message is described in the target-required format. The adapter's main role is at run-time. Grounding rules executed by adapter are specified at design-time. Along with message transformations, the adapter will also create a request (goal) describing services to be invoked at the receiving ends. These goals will later be resolved by the semantically-enabled middleware during process mediation. The main components within the Adapter Framework are:

The Communication Manager is a gateway to the adapter framework which can be accessed by MessageListener via a Web-service interface or TCP/IP port (for non-Web Service EPRs). The Message Listener waits for any new message incoming from the network. Once requested by a back end application, the Communication Manager interprets the request and invokes either the Adapter Manager or the Update Manager. If the invocation is successful, it returns a response message which contains either a 'success' message or the 'response' returned from the target application.

The Adapter Manager is the core component of the Adapter Framework which schedules job between adapters and the data format Validator. It updates and refreshes the list of the adapters as they are registered and unregistered.

The Validator is responsible for validating the data format as claimed by the sending application. An XML message, for example, sent by back end application, should pass the XML data format validation process.

The Adapter Pool is a "repository or container" of all the registered adapters.

The XML2WSML adapter is a specialized adapter which transforms (lifts) the XML message (syntactic) to Web ser-

[7]http://www.x12.org/
[8]http://www.interfaceware.com/manual/llp.html
[9]http://www.whb.ie/OurServices/AcuteHospitalServices

vice modeling language(WSML[10]) (semantic).

The WSML2XML adapter is a specialized adapter which transforms (lowers) the WSML (semantic) to XML message (syntactic).

The Database Manager manages database connection, exceptions, and utilities.

Semantics-Enabled Middleware (WSMX)[11]: A detailed description of WSMX, WSML, and Web service modeling tool (WSMT[12]) is outside the scope of this paper [9, 4, 6], therefore we explain WSMX and WSML from the PPEPR perspective, mainly focusing on data and process meditation. Healthcare standards define a set of messages and functionalities which may not match across standards or systems. These functionalities are exposed as Web services and the first step towards semantic Service-Oriented Architecture (sSOA) is to create semantic descriptions of Web services (during design-time), listing 1 shows the snippet of WSML Web service description of HL7 v3 order fulfiller Web service.

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml−syntax/wsml−full"
namespace { _"http://host:port/OrderFulfillerWebSevice.wsml#" ,
      wsmostudio _"http://www.wsmostudio.org#",
      dc _"http://purl.org/dc/elements/1.1/",
      FO _"http://host/development/ontlogies/v2FunctionalOntlogoy.
            wsml",
      wsml _"http://www.wsmo.org/wsml/wsml−syntax#",
      MO _"http://host/development/ontlogies/POLB_IN0021200.wsml#
            "}

webService OrderFulfillerWebService
      importsOntology MO#POLB_IN002120
      usesMediator _"http://host:WSMX/ooMediator"

capability OrderFulfillerCapability
     precondition
               nonFunctionalProperties
                    dc#description hasValue "Laboratory Observation Order
                           Activate, Fulfillment Request with message
                           transmission wrapper and payload."
               wsmostudio#version hasValue "0.5.4" _"http://host#
                    v2FunctionalOntlogoy" hasValue FO#
                    OrderFulfillerWebServiceOntology
               endNonFunctionalProperties
               definedBy
     ?mesasge memberOf MO#OrderActivateFulfillmentRequest and
     MO#controlActProcess(?message,?controlActProcess) and
               ...................
interface OrderFulfillerInterface

choreography OrderFulfillerChoreography
     stateSignature OrderFulfillerStatesignature
transitionRules OrderFulfillerTransitionRules if (?request
memberOf OrderActivateFulfillmentRequest) then
            add(_#1 memberOf OrderActivateFulfillmentRequest_Ack)
            endIf
            .....
orchestration OrderFulfillerWebServiceOrchestration
     in MO#OrderActivateFulfillmentRequest
     withGrounding { _"http://host/OrderFulfillerWebSevice.wsdl#wsdl.
            interfaceMessageReference(OrderFulfillerInterface/OrderFulfill/
            in0)"}
     out MO#OrderActivateFulfillmentRequest_Ack
     withGrounding { _"http://host/OrderFulfillerWebSevice.wsdl#wsdl.
            interfaceMessageReference(OrderFulfillerInterface/OrderFulfill/
            out0)"}
     ......
forall {?request} with
         (?request memberOf MO#OrderActivateFulfillmentRequest )
     do
            add(_#1 memberOf MO#OrderActivateFulfillmentRequest_Ack
                         )
  endForall ........
```

Listing 1: WSML Web service description of the order fulfiller Web service

In listing 1(top to bottom), the namespace section describes the identifiers(IRIs, sQNames)[4] for message ontology(MO) and functional ontology(FO). The usesMediator keyword identifies the mediator (e.g. ooMediator[4]) which connects different Goals, Web Services and Ontologies, and enable inter-operation by reconciling differences in representation formats, encoding styles, business protocols, etc. In capability section, preconditions describe the condition over information space state before the execution of a Web Service. For example, non-functional properties describes the type of message accepted as input and corresponding HL7 v2 OrderFulfiller Web Service(e.g. HL7 v2 and HL7 v3 complaint EPRs interactions). Interface describe how to interact with a service from the requester point of view (choreography) and how the service interacts with other services and goals it needs to fulfill in order to fulfill its capability (orchestration), which is the provider point of view. The OrderFulfillerInterface describe transition rules as a part of choreography, for example, request and corresponding Acknowledgment(response) messages. In orchestration section, the input and output messages and respective WSDL grounding is defined, for example, wsdl.interfaceMessageReference (ServiceInterface/ Operation/input or output message)[13]

The listing 1 describes only two messages (OrderActivateFulfillmentRequest, OrderActivateFulfillmentRequest_Ack) , mediator used, choreography, and orchestration associated with these messages. The complete message exchange pattern is shown in Figure 2 and dotted lines within listing 1 indicates the description of all the messages exchanged by orderfulfiller Web service. The WSML web service description are stored in semantically-enabled repository (e.g WSMO4RDF[14] repository within ORDI SG framework) , WSMX picks-up these service descriptions automatically during discovery, composition, choreography, and orchestration of Web services. The WSML Web service descriptions are created manually during each setup of existing and/or new EPRs, which is a major integration task for HL7 complaint EPRs within PPEPR integration platform.

The WSMX mediation (data and process) components have three main functions; one used during the design time and the other two used during runtime. First, a set of mappings is created by domain experts (e.g. healthcare professionals), using a mapping tool - WSMT. Second, the "Mapping Rule Generator" within mediation components is called internally during runtime to transform the mappings into mapping rules. Mappings express the similarities between the two ontologies, while the mapping rules describe how these similarities are used in order to transform instances of the first ontology in instances of the second ontology. Mapping rules express the mappings in an executable way. The third and final step of the mediation process is the execution of the mapping rules. These mapping rules are received from the "mapping rules generator" and executed inside the WSMX rule execution environment[3].

[10]http://www.wsmo.org/wsml/

[11]http://www.wsmx.org/

[12]http://sourceforge.net/projects/wsmt

[13]http://www.wsmo.org/TR/d24/d24.2/v0.1/

[14]http://www.ontotext.com/ordi/ORDI_SG/Wsmo4rdf.html

5. ONTOLOGY DEVELOPMENT

Ontology development for PPEPR includes message ontologies to represent the information sent between the components and functional ontologies to represent the ordering of relations in the execution of the message exchange patterns:

5.1 Message ontology

The ontology of a healthcare standard is developed from the specification provided in the XML Schema format. This is done by processing grounding rules defined by the developers that transform the HL7 XML Schema to an ontological representation (Bi-directional). Heterogeneities at the ontological level are identified and mapping definitions between ontologies are created using an ontology modelling tool WSMT. Both Grounding and Mapping definitions at the conceptual level are stored and used by the adapter and semantic enabled middleware at run-time.

```
<POLB_IN002120.Message>
  <controlActProcess>
    <subject>
      <LabTestOrder
        <author>
          <orderer>
            <id displayable="true" extension="1−976−245"
               root="2.16.840.1.113883.19.3.2409"/>
            <healthCareProviderOrderer>
              <name use="L">
                <given>Dr. Rise</given>
              </name>
            </healthCareProviderOrderer>
          </orderer>
        </author>
      </LabTestOrder>
    </subject>
  </controlActProcess>
</POLB_IN002120.Message >
```

Listing 2: XML Message (Simplified version): HL7 v3 laboratory observation order activates fulfillment request

Listing 2 shows a simplified XML message from a HL7 v3 compliant EPR and Listing 3 shows the equivalent WSML instance. A detailed explanation of the HL7 message structure and the XML to WSML transformation process are outside the scope of this paper. Therefore we will briefly explain the syntactic (XML) and semantic (WSML) relationship of both the messages. The message POLB_IN002120 (HL7 v3) is a "laboratory observation order activate fulfillment request" message. This interaction message is created when a "fulfillment request" is communicated between the "order placer" and the "fulfiller" (e.g. Figure 2). Grounding Rules(e.g. XSLTs[15]) are applied to transform the POLB_IN 002120 xml message to the WSML instance. For example, the "controlActProcess" tag in the XML message is equivalent to the "control_act_process" relation within the WSML instance. Similarly, the "given" tag of the XML message is equal to the "given" relation instance. Thus, the "Dr. Rise" value in the "given" tag transforms into the second parameter of the "given" relation instance. Instances of WSML concepts are used in relation instance parameters to link the different relation instances. This is similar to the way complex types in XML Schema maintain a nested form within an XML message.

[15]http://www.w3.org/TR/xslt

```
_"http://host/development/ontologies/POLB_IN002120.wsml#"

instance polb_in002120_message memberOf POLB_IN002120.Message
relationInstance control_act_process
controlActProcess(polb_in002120_message, message_controlActProcess)
instance message_controlActProcess memberOf
  POLBIN002120ControlActProcess
relationInstance subject subject(message_controlActProcess,
  message_subject)
instance message_subject memberOf POLBIN002120Subject
relationInstance lab_test_order LabTestOrder(message_subject,
  message_labTestOrder)
instance message_labTestOrder memberOf LabTestOrder
relationInstance author author(message_labTestOrder, message_author)
instance message_author memberOf Author
relationInstance orderer orderer(message_author, message_orderer)
instance message_orderer memberOf Orderer
relationInstance id id(message_orderer, ii)
instance ii memberOf II
relationInstance displayable displayable(ii, _boolean("true"))
relationInstance extension extension(ii, "1−976−245")
relationInstance root root(ii, "2.16.840.1.113883.19.3.2409")
relationInstance healthCareProviderOrderer
  healthCareProviderOrderer(message_orderer, message_providerOrderer)
instance message_providerOrderer memberOf ProviderOrderer
relationInstance name
  ProviderOrderer_name(message_providerOrderer, en)
instance en memberOf EN
relationInstance given given(en, "Dr. Rise")
relationInstance use use(en, "L")
```

Listing 3: WSML Instance (Simplified version): HL7 v3 laboratory observation order activates fulfillment request.

WSML uses similar XPath[16] patterns to represent "instance" as well as "relationInstance" of concept and the relation, therefore it looses the structure of original XML message. To resolve this issue during the lowering process, we use XPath's context node feature to uniquely identify instance, relation instance and attribute values. For example, the "given" relationInstance (see second last line listing 3) and its first parameter "en" are pointed to by the contexts "//wsml: relationInstance [wsml:memberOf = 'given']" and "/wsml: parameterValue [wsml:parameter= 'parameter1']/wsml: value" respectively [see listing 4].

```
<xsl:template match="/">
<xsl:for−each select="//wsml:relationInstance[wsml:memberOf = 'given
      ']">
<given>
<xsl:value−of select="./wsml:parameterValue[wsml:parameter= '
      parameter1']/wsml:value"/>
</given>
</xsl:for−each>
</xsl:template>
```

Listing 4: XSLT (Simplified version): Lowering WSML Instance(Listing 2) to XML message (Listing 1)

5.2 Functional ontology

HL7 categorises healthcare events that are used to annotate healthcare functionalities. Similar to standards in other domains, such as RosettaNet[17] for Supply Chain Management in ICT, HL7 not only defines the message content, but also the business logic to achieve certain functionality in the health care domain. Figure 2 shows the choreography

[16]http://www.w3.org/TR/xpath

[17]http://www.rosettanet.org/

(message exchange patterns) between the order placer and the order fulfiller whereas the activity diagram in figure 3 shows the process model of the order placer to achieve the actual healthcare process. It is sufficient if both parties, the process placer and the process fulfiller model and execute a process according to the message exchange patterns defined in HL7 and shown in the left and right part of figure 2.

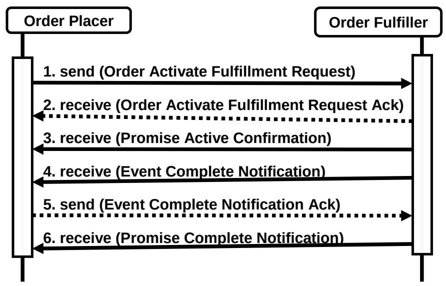

Figure 2: Interaction Diagram in HL7 Fulfillment Request

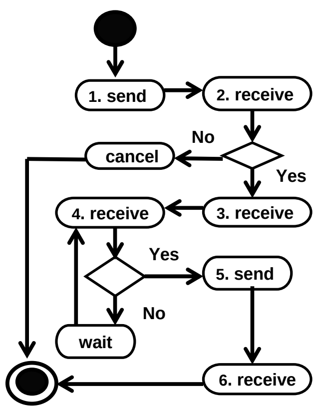

Figure 3: Business Process of Order Placer in a HL7 Fulfillment Request

To model and execute such message exchange patterns, it is necessary to employ a process modelling and execution standard which is able to reference ontological elements and allow their mapping within the model. BPEL4SWS[18][5] a conservative set of language extensions to BPEL gives us the possibility to reference ontological elements within a business process description.

BPEL4SWS defines a new mechanism to describe the communication between two partners without dependency on WSDL which allows us to include goals in the execution process. BPEL4SWS introduces a new element, the `<b4s:conversation>` element, independent from WSDL. This element enables grouping of interaction activities and thus enables defining a complex message exchange between two partners.

Interaction activities in BPEL4SWS link messages to goals. The interaction activities are grouped in a conversation which allows us to link to a Web Service Modeling Ontology (WSMO)[19] Goal[5]. The WSMO Goal describes what the process expects from a service that plays the partner role in this particular interaction by means of the requested capabilities.

Listing 5 shows a snippet of the BPEL4SWS document to describe the internal process including the interface behaviour of the order placer as depicted in figure 3.

```
<b4s:conversations>
  <b4s:conversation b4s:name="OrderPlacer" b4s:goalURI="http://host
        /ontologies/HL7v3FuncOntology#goalURL1" />
  <b4s:conversation b4s:name="OrderFullfiller" b4s:goalURI="http://
        host/development/ontologies/HL7v3FuncOntology#goalURL2"
        />
</b4s:conversations>

<sequence sa:modelReference="http://host/ontologies/
      HL7v3FuncOntology#actEventCompleteNotification">
  <receive name="EventCompleteNotification" sa:modelReference="http
        ://host/ontologies/HL7v3FuncOntology#
        EventCompleteNotification" partnerLink="OrderFullfiller"
        portType="spwsdl:OrderFullfillerPortType" operation="
        requestNotification" variable="Notification" createInstance="yes
        " />

  <extensionActivity>
    <b4s:invoke name="invokePromiseCompleteNotificationOp"
          modelReference="http://host/ontologies/HL7v3FuncOntology
          #actEventCompleteNotificationAck" b4s:inputVariable="
          NotificationAck" b4s:outputVariable="
          PromiseCompleteNotification" b4s:conversation="
          FulfillmentRequest" />
  </extensionActivity>
</sequence>
```

Listing 5: Snippet of BPEL4SWS according to HL7 Fulfillment Request

6. EPR INTEGRATION: PPEPR

Healthcare is a complex domain, comprising vendors, standards, legacy systems, and information systems which inherently differ from one another. PPEPR provides a unique approach to interoperability. The core solution lies in enabling semantic interoperability between existing and new EPR systems. The PPEPR architecture considers three types of integration between EPRs based on their web service capabilities (or lack thereof).

Type 1: Existing EPR [non-Web services] <==> Existing EPR [non-Web services]

This type of interaction is focussed on existing EPRs, which are mostly HL7 v2.x based. At this level EPRs will be able to exchange messages in EDI[20] and XML format. The majority of current HL7 v2.x based systems do not use Web services, and their immediate requirement is to exchange messages. The PPEPR adapter can interact bi-directionally with these EPRs at a specified TCP/IP port.

Limitations: The goal of PPEPR is to achieve the maximum level of interoperability between existing and/or new systems, without upgrading existing or introducing new EPR systems. Even though HL7 v2.x has categorized the events in healthcare by considering service functionality which reflects the business logic in this domain, implementations are not service-oriented. These service functionalities could be used to deploy Web services by upgrading existing EPR systems to Web service enabled EPR. This transition, although

[18] http://lists.w3.org/Archives/Public/public-ws-semann/2007Jun/0000

[19] http://www.wsmo.org/

[20] http://en.wikipedia.org/wiki/UN/EDIFACT

outside the scope of PPEPR, would enable those systems to benefit from advanced interoperability.

Type 2: Existing EPR [non-Web services] <==> Web-Service enabled EPR

This type of integration is the most complex (e.g. HL7 2.x <==> HL7 v3), since EPRs (non-Web services) are required to communicate with the other EPRs (Web-services) via SOAP.

Limitations: As, one side of the EPR integration process does not employ Web services the Web service Enabled EPR cannot invoke the services directly. PPEPR's Adapter can receive the message at a TCP/IP port, converting the message to XML if the received message is via EDI, creating a SOAP message, then invoking the EPR at the other end, after mediation. This transformation and routing mechanism is defined at design time. Therefore, two or more EPRs at this level should be aware of each other, before they interact. Also, this type of integration cannot enjoy the interoperability advantages (e.g. discovery, composition, orchestration, etc.) of semantic Web services at the service and/or process level but achieves the major goal of sharing healthcare messages.

Type 3: Web-Service enabled EPR (1) <==> Web-Service enabled EPR (2)

This type of integration in PPEPR architecture offers the best interoperability solution by achieving syntactic as well as semantic interoperability.

7. RELATED WORK

PPEPR approach is based on semantic Service-Oriented Architecture (sSOA) and other projects which employs the similar principal are:

COCOON[21] is a 6th Framework EU integrated project aimed at setting up a set of regional semantics-based healthcare information infrastructure with the goal of reducing medical errors. In order to enable a seamless integration of eHealth services, Semantic Web Services technology is applied.

ARTEMIS[22] is a STREP project supported in the 6th Framework by the European Commission. ARTEMIS aims to develop a semantic Web Services based Interoperability framework for the healthcare domain. Artemis has a peer-to-peer architecture in order to facilitate the discovery and consumption of healthcare web services.

RIDE[23] and SemanticHEALTH[24] projects are E.U road map projects with Special Emphasis on Semantic Interoperability. The road map will be based on consensus of the research community, and validated by stakeholders, industry and Member State health authorities.

The major differences between eHealth projects described above and our approach are:

- Other projects impose constraints on the EPRs (e.g. transition from traditional to Web services) but our aim is to integrate them without imposing any constraint on existing or proposed EPRs.
- None of the other projects provide an integration solution between non service-oriented and service-oriented (Web services).
- PPEPR applies a new mechanism to describe the communication between two partners without a dependency on WSDL. As described in section 5.2, BPEL4 SWS introduces a new element, (`<b4s:conversati on>`), which is not dependant on a `<partnerLink Type>` and as such is not dependant on WSDL. This element enables the grouping of interaction activities and thus enables defining a complex message exchange between two partners. For example, Artemis introduced business process template "BP template" to mod el business process at design time. BP template is dependent on WSDL (e.g. `<partnerLinkType>`) to describe a contract between two partners in terms of roles and corresponding WSDL portTypes. Also, for interaction activities "BP template" mainly relies on partnerLink and a WSDL operation.

8. CONCLUSIONS

Current healthcare enterprise applications need greater flexibility and scalability to meet the challenges of heterogeneity of healthcare systems at all levels - data, process, services, etc. The architecture of any integration system holds the key to offer a dynamic, flexible and scalable solution. As we have discussed above, healthcare is a complex domain. Actors include vendors, standards, legacy systems, and information systems all of which must interoperate to provide healthcare services. PPEPR provides an interoperability solution without imposing any constraint on existing or proposed EPRs. The major advantage of PPEPR approach is the "Plug and Play" feature, which requires minimal configuration during setup. Within scope of PPEPR, EPRs are categorized based on international healthcare standards: (1) HL7 v2.x, (2) HL7 v3, and (3) CEN TC251 13606/openEHR. These three categories of standards are quite different in their overall approach towards an EPR, but share many semantic similarities, especially the Reference Information Models (RIM) of HL7 v3, and CEN TC251 13606/openEHR. These similarities are the linchpin of ontology development and mapping.

We have presented two major aspects of PPEPR: ontology specification and adapter framework development. Each ontology specified as part of this project is further categorised into a message and a functional ontology. A functional ontology describes the semantics required for interaction between EPR systems, based on the interaction events within healthcare standards to provide domain based workflows between interactive EPR systems. A message ontology explicitly defines the semantics associated with healthcare messages required to be exchanged for data interoperability between EPR systems. Differences in these specifications are resolved by semantic mapping techniques.

The adapter framework receives the grounding rules developed at design-time and plays a crucial role in integrating EPRs at run-time. The adapter framework follows a similar design pattern to that of the Java Connector Architecture (JCA[25]). Similarity in features include component based development and deployment, flexibility in deployed adapters, and automated data format validation.

The first prototype of PPEPR has successfully integrated HL7 v3 and HL7 v2.x within an experimental environment;

[21] http://www.cocoon-health.com/

[22] http://www.srdc.metu.edu.tr/webpage/projects/artemis

[23] http://www.srdc.metu.edu.tr/webpage/projects/ride/

[24] http://www.semantichealth.org/

[25] http://java.sun.com/j2ee/connector/

Future developments include mapping and mediating between HL7, CEN TC251 13606, and openEHR standards. PPEPR will be validated in conjunction with end user clinicians in leading Irish Hospitals. In the first prototype of PPEPR we have not yet covered all integration issues (e.g. security) during adapter development. However, since we employ component based development, future components can be easily integrated. Also, this paper focuses on, introducing PPEPR architecture, data and process mediation. Other service-oriented features such as publishing, discovery, etc. are outside the scope of this paper.

9. ACKNOWLEDGMENTS

We would like to thank Brahmananda Sapkota, Armin Haller, and James Cooley, for their comments and input to this document. This material is based upon works supported by the Science Foundation Ireland project Lion under Grant No.(SFI /02/CE1/I131) and by Enterprise Ireland under Project PPEPR (CFTD 2005 INF 224).

10. REFERENCES

[1] M. L. Brodie, C. Bussler, J. de Bruijn, T. Fahringer, D. Fensel, M. Hepp, H. Lausen, D. Roman, T. Strang, H. Werthner, and M. Zaremba. Semantically enabled service-oriented architectures: A manifesto and a paradigm shift in computer science. Technical Report TR20051226, DERI, 12 2005. Available at `http://www.deri.ie/fileadmin/documents/DERI-TR-2005-12-26.pdf`.

[2] C. Bussler, D. Fensel, and A. Maedche. A Conceptual Architecture for Semantic Web enabled Web Services. *SIGMOD Rec.*, 31(4):24–29, 12 2002.

[3] E. Cimpian and A. Mocan. WSMX Process Mediation Based on Choreographies. In *1st International Workshop on Web Service Choreography and Orchestration for Business Process Management*, Nancy, France, 9 2005. IEEE Computer Society.

[4] J. de Bruijn, H. Lausen, A. Polleres, and D. Fensel. The Web Service Modeling Language WSML: An Overview. In *Proceedings of the 3rd European Semantic Web Conference (ESWC 2006)*, volume 4011 of *Lecture Notes in Computer Science, LNCS*. Springer, 6 2006.

[5] A. Filipowska, A. Haller, M. Kaczmarek, T. V. Lessen, J. Nitzsche, and B. Norton. Process Ontology Language and Operational Semantics for Semantic Business Processes. BPEL4SWS specification, Available at `http://www.ip-super.org/res/Deliverables/D1.3.pdf`.

[6] M. Kerrigan. The WSML Editor Plug-in to the Web Services Modeling Toolkit. In *Proceedings of the 2nd WSMO Implementation Workshop (WIW)*, 6 2005.

[7] Sun Microsystems Inc,. Implementing Health Information Technology for RHIO Success. A Sun White Paper, Available at `http://www.sun.com/software/whitepapers/integration_suite/rhio_healthercare_wp.pdf`.

[8] E. D. Valle, D. Cerizza, P. D. M. Veli, B. Yildirak, K. Gokce, B. Laleci, and H. Lausen. The Need for semantic Web Service in the eHealth, 6 2005. In W3C Workshop-SWSF, Innsbruck, Austria, Position paper.

[9] T. Vitvar, A. Mocan, M. Kerrigan, M. Zaremba, M. Zaremba, M. Moran, E. Cimpian, T. Haselwanter, and D. Fensel. Semantically-enabled service oriented architecture: Concepts, technology and application. *Service Oriented Computing and Applications*, 5 2007.