

ReMINE

High performances prediction, detection and monitoring
platform for patient safety risk management

FP7 Contract: 216134



– Deliverable –

D.3.3 First Revision Data & Process model system framework

Document Information**Document Name:** ReMINE_D3.3_IW_WP3_10-03-16_Final**Revision:** V0.7**Revision Date:** 15/03/2010**Author:** Adrian Podea (Info World)**Contributors:** ICCS, INDRA**Security:** Confidential (Consortium Only)**Document Information**

Consortium, European Commission

Approvals

	Name	Beneficiary	Date	Visa
<i>Project Coordinator</i>	Michele CARENINI	NOEMALIFE		
<i>Technical Manager</i>	Fernando GUMMA	NOEMALIFE		

Document History

Revision	Date	Modification	Author
Version 0.1	10/02/2010	ToC first draft	IW
Version 0.2	20/02/2010	Executive Summary , Chapter 2.1 Chapter 3: Architecture	IW
Version 0.3	22/02/2010	Chapter 4.3: Master Patient Index Service	IW
Version 0.4	24/02/2010	Chapter 4.2: TOS Enhancements	IW
Version 0.5	25/02/2010	ICCS contribution on chapter 5	ICCS
Version 0.6	12/03/2010	ICCS & INDRA contributions	ICCS & INDRA

Version 0.7	15/03/2010	IW final revision	IW
-------------	------------	-------------------	----

1. Executive summary.....	6
2. Introduction.....	6
2.1. Deliverable vs. Project objectives	6
2.2. Deliverable structure and Partners contribution	7
3. Architecture overview	8
4. Metadatabase enhancements.....	9
4.1. Taxonomy and Ontology Service enhancements.....	9
4.1.1. Business scenarios and use cases	9
4.1.2. Service contracts.....	10
4.1.3. Detailed functional model on the extended interface	11
4.1.4. Data contracts.....	14
4.1.5. Fault contracts	18
4.1.6. Database layer	19
4.2. Master Patient Index Service	22
4.2.1. Architectural overview.....	22
4.2.2. Business scenarios and use cases	23
4.2.3. Service contracts.....	26
4.2.4. Detailed functional model on each interface	27
4.2.5. Data contracts.....	33
4.2.6. Fault contracts	42
4.2.7. Database layer	45
4.2.8. Service metadata management.....	46
5. Process mapper	49
5.1. General Description	49
5.2. Process Mapper for Sacco.....	50
5.2.1 User interface	62

5.2.2 Data Mapper.....	70
5.3. Process Mapper for Kauhajoki	71
5.3.1 User interface	72
5.3.2 Data Mapper.....	76
5.3.3 RFID integration.....	79
5.4. Process Mapper for Niguarda	86
5.4.1 Introduction.....	86
5.4.2 Overview.....	86
5.4.3 Correlation	87
5.4.4 “NiguardaMsgSortingMain” model	87
5.4.5 “PatientAdmission” model	88
5.4.6 “NeurologyAssessment” model.....	88
5.4.7 “CerebralCTScanManagement” model	89
5.4.8 “LabExam” model	90
5.4.9 “ECGandAdministrationOfReperfusion” model	93
5.4.10 “ReAssessment” model	94
5.5. Database (RLUS) Connection	95
5.5.1 Retrieving Patient Information from RLUS.....	95
5.5.2 Updating Patient Information in RLUS	96
5.6. Business rules Engine (BRE) integration.....	97
6. Conclusion	100
6.1. Summary	100
6.2. Next steps and activity plan	100
7. Glossary	101

1. Executive summary

Deliverable D.3.3 First Revision Data & Process Model system framework describes the components that ensure data modeling, storage and retrieval, as well as the process mapper components for the Kauhajoki, Sacco and Niguarda scenarios. As regarding the data issues, the deliverable describes several enhancements that were made on the metadatabase component. The enhancements consist in extending the Taxonomy and Ontology Service, and also in the introduction of the Master Patient Index Service. The TOS service is fully described in Deliverable D3.5 First Revision of the WP3 framework, so the present deliverable focuses only on how the improvements made affect the previous service definitions. The services are presented by specifying: architectural overview, business scenarios and use cases, detailed functional model on each service interface, as well as the detailed design model with information about the service contracts, data contracts, fault contracts and the database layer. The main sections from the present deliverable are outlined below:

- **Architecture overview** describes the main components and the relationships between them with an architectural diagram
- **Taxonomy and Ontology Service enhancements** details the extensions made to the service API that offers access to extensible terminological resources. The improved TOS API enables the retrieval of concepts with custom properties, defined on the ontology and brings an even more powerful modeling of semantic information within TOS.
- **Master Patient Index Service** provides the definition of the web service that realizes the patient management within the ReMINE project. Additionally, the service used for metadata management is also documented in the chapter ending.
- **Executable models and process mapper** describes how the involved processes are modeled and executed, with specific case studies for the Kauhajoki, Sacco and Niguarda scenarios.

2. Introduction

2.1. Deliverable vs. Project objectives

Deliverable D3.3 First Revision Data & Process model system framework is the first one out of the three deliverables regarding Task 3.1 – Data & Process advanced store tool. This deliverable addresses the first steps in the development of the technological infrastructure for storing inside the metadatabase in a semantic format all information that cannot be acquired directly from the WP2. Task 3.1 objectives are presented below.

Sub-Task.3.1.1 Semantic identification – objectives:

- Identification of the semantic structure to store the entire set of data arriving from WP2 (Data Capture and RAPS alerting) and WP1 (User requirement, Patient Care processes management modeling)
- Definition and developing of the semantic API interface layer to and from the Metadatabase
- Usage and access of External data, Rules and process definition

Sub-Task.3.1.2 Information acquisition and storage (DIS) – objectives:

- Developing a tool that using BPM process allow not technical user to input and store inside the Metadatabase all the information that are in paper format, human historical process (people common and ordinary process), oral procedures, de-facto rules. All this information will be stored using a semantic metadata structure and organization.
- Developing all the API interface need to connect this module with other REMINE module in order to exchange data and process information

2.2. Deliverable structure and Partners contribution

Chapters	Responsible partner
1. Executive summary	ALL
2. Introduction	IW
3. Architecture Overview	IW
4. Metadatabase enhancements	
4.1. Taxonomy and Ontology Service enhancements	IW
4.2. Master Patient Index Service	IW
5. Executable models and process mapper	
5.1. General Description	ICCS
5.2. Process Mapper for Kauhajoki	ICCS
5.3. Process Mapper for Sacco	INDRA
5.4. Process Mapper for Niguarda	INDRA
6. Conclusion	ALL

3. Architecture overview

Figure 1 represents the WP3 place in the data-flow architecture of the whole ReMINE project. WP3 consists of three different components: the Process Mapper, the Database (database service with reasoner) and Data Mining & Knowledge Inference.

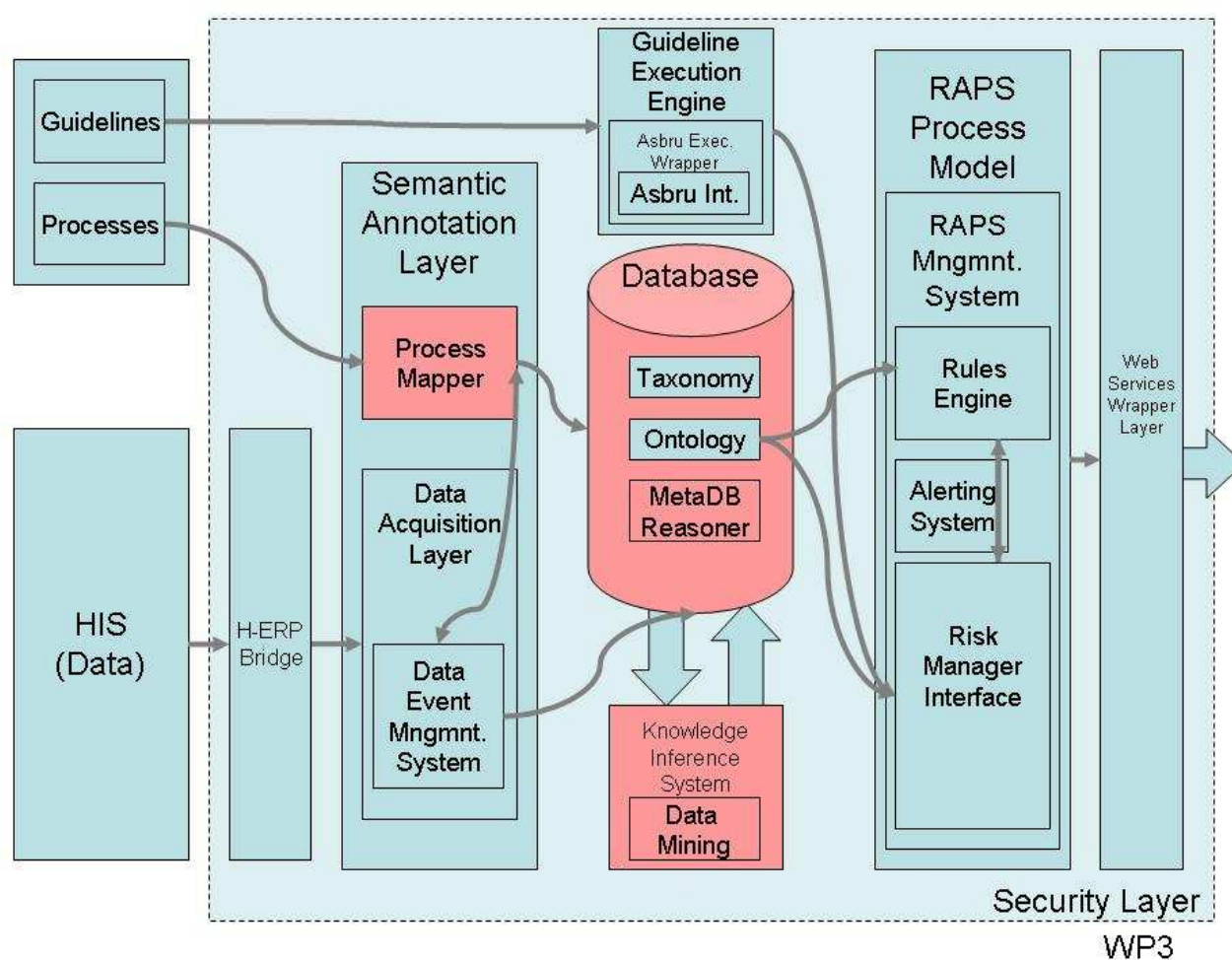


Figure 1 – The place of WP3 in ReMINE architecture

WP3 consists of the communication between the Data Acquisition Layer, the RLUS repository which is accessible through the exposed RLUS API, the additional available services (TOS and Security Service), the Metadatabase Reasoner and the Data Mining and Knowledge Inference component, where the latter exposing in turn its results through an API for further utilization. Another new service is represented by the Master Patient Index service (MPI) used for patient management within the ReMINE project.

4. Metadatabase enhancements

4.1. Taxonomy and Ontology Service enhancements

The taxonomy and ontology service (TOS) facilitate the access to external terminological resources using a HL7 CTS1 (Common Terminology Services v1) implementation. The full specification for TOS is available in the related deliverable D3.5 First Revision of the WP3 framework, including detailed information about: service architecture, business scenarios, Message API - including service contracts, data contracts and fault contracts -, and the database layer.

For realizing Task 3.1, new features are required in TOS for retrieving enhanced information about the terminological resources. The additional features are described in the present chapter in terms of business scenario, service contract, detailed functional model, data contracts and fault contracts.

The TOS enhancements involve enabling concepts to encapsulate more semantic information through concept properties. The original concepts retrieved using TOS were conforming to a standard schema of information. The schema was extended to accept custom properties for concept modeling the additional semantic information available. As a result, the Message Layer Browser service contract was transformed to enable the retrieval of concepts associated with certain requested properties.

4.1.1. Business scenarios and use cases

The actors implicated in the TOS use cases are: TOS Consumer, TOS (Service) and the TOS Database Agent on the Database layer. Below, the sequence diagram describes the system interactions:

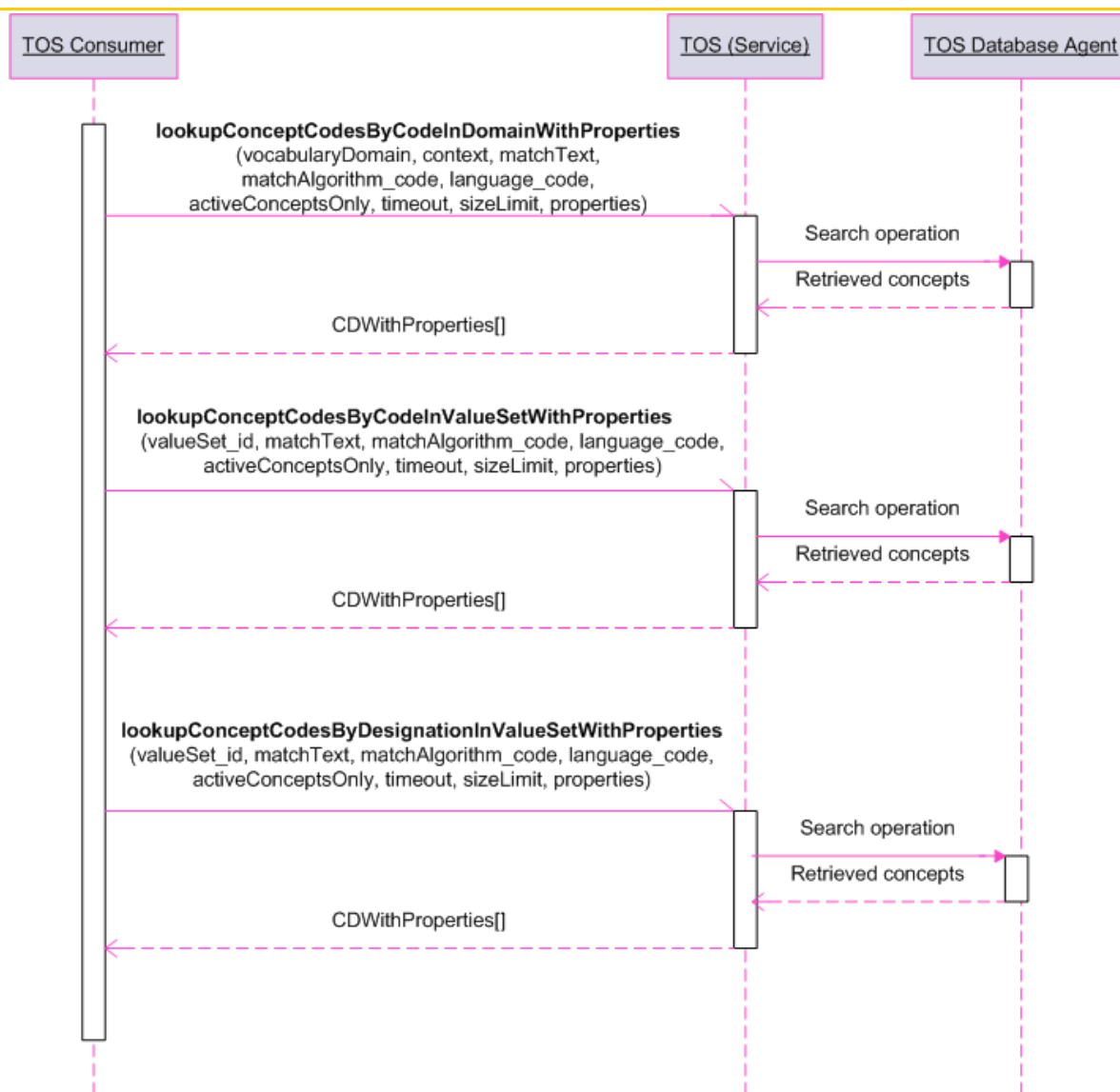


Figure 2 – Sequence diagram for primary uses cases regarding TOS enhancements

4.1.2. Service contracts

The Message Layer Browser (*CTSMAPI.BrowserOperations*) specifies functions which retrieve all supported vocabulary domains, code systems and value sets; it also looks up for these CTS concepts.

The service contract of the Message Layer Browser is enhanced with additional operations regarding retrieval of concepts with properties:

- lookupConceptCodesByCodeInDomainWithProperties
- lookupConceptCodesByCodeInValueSetWithProperties
- lookupConceptCodesByDesignationInValueSetWithProperties

4.1.3. Detailed functional model on the extended interface

Message Layer Browsing Functions (CTSMAPL.BrowserOperations)

The following functions were added to the Message Layer Browsing Functions to enhance the capabilities of the Taxonomy and Ontology Service:

- lookupConceptCodesByCodeInDomainWithProperties

Business friendly name	Retrieve concept codes with properties (CDWithProperties array)
Signature	<i>CDWithProperties[]</i> lookupConceptCodesByCodeInDomainWithProperties(<i>string</i> vocabularyDomain_Name, <i>string</i> context, <i>string</i> matchText, <i>string</i> matchAlgorithm_code, <i>string</i> language_code, <i>bool</i> activeConceptsOnly, <i>int</i> timeout, <i>int</i> sizeLimit, <i>List<string></i> properties);
Description	Retrieves an array of concepts that are matched in a vocabulary domain. The search is done by code. Concepts have additional properties as requested in the parameter <i>properties</i> .

Inputs	<p>Vocabulary domain to search in</p> <p>Application Context</p> <p>Text to match</p> <p>Matching algorithm (StartsWithIgnoreCase, EndsWithIgnoreCase, ContainsPhraseIgnoreCase, IdenticallyIgnoreCase)</p> <p>The language used for the matched text</p> <p>Whether it searches only active concepts</p> <p>Timeout in seconds</p> <p>Maximum number of concepts to return (0 = no limit)</p> <p>The list of properties to be returned for each concept</p>
Outputs	Matched concepts as list.
Exceptions	<p><i>UnexpectedError</i></p> <p><i>UnknownMatchAlgorithm</i></p> <p><i>BadlyFormedMatchText</i></p> <p><i>TimeoutError</i></p> <p><i>UnknownVocabularyDomain</i></p> <p><i>NoApplicableValueSet</i></p> <p><i>UnknownLanguageCode</i></p>

- lookupConceptCodesByCodeInValueSetWithProperties

Business friendly name	Retrieve concept codes with properties (CDWithProperties array)
Signature	<p><i>CDWithProperties[]</i></p> <p>lookupConceptCodesByCodeInValueSetWithProperties(<i>string</i> valueSet_id, <i>string</i> matchText, <i>string</i> matchAlgorithm_code, <i>string</i> language_code, <i>bool</i> activeConceptsOnly, <i>int</i> timeout, <i>int</i> sizeLimit, <i>List<string></i> properties);</p>

Description	Retrieves an array of concepts that are matched in a value set. The search is done by code. Concepts have additional properties as requested in the parameter <i>properties</i> .
Inputs	Value set's ID to search in Text to match Matching algorithm (StartsWithIgnoreCase, EndsWithIgnoreCase, ContainsPhraseIgnoreCase, IdenticallyIgnoreCase) The language used for the matched text Whether it searches only active concepts Timeout in seconds Maximum number of concepts to return (0 = no limit) The list of properties to be returned for each concept
Outputs	Matched concepts as list.
Exceptions	UnexpectedError UnknownMatchAlgorithm BadlyFormedMatchText TimeoutError UnknownValueSet UnknownLanguageCode

- lookupConceptCodesByDesignationInValueSetWithProperties

Business friendly name	Retrieve concept codes with properties (CDWithProperties array)
Signature	<i>CDWithProperties[]</i> lookupConceptCodesByDesignationInValueSetWithProperties (<i>string</i> valueSet_id, <i>string</i> matchText, <i>string</i> matchAlgorithm_code, <i>string</i> language_code, <i>bool</i> activeConceptsOnly, <i>int</i> timeout, <i>int</i> sizeLimit, <i>List<string></i> properties);

Description	Retrieves an array of concepts that are matched in a value set. The search is done by description (designation, textualPresentation). Concepts have additional properties as requested in the parameter <i>properties</i> .
Inputs	Value set's ID to search in Text to match Matching algorithm (StartsWithIgnoreCase, EndsWithIgnoreCase, ContainsPhraseIgnoreCase, IdenticallyIgnoreCase) The language used for the matched text Whether it searches only active concepts Timeout in seconds Maximum number of concepts to return (0 = no limit) The list of properties to be returned for each concept
Outputs	Matched concepts as list.
Exceptions	UnexpectedError UnknownMatchAlgorithm BadlyFormedMatchText TimeoutError UnknownValueSet UnknownLanguageCode

4.1.4. Data contracts

The precedent version of TOS made use of simpler concept descriptors with a limited range of attributes. However, the necessity that concepts have additional properties emerged, and the current TOS enhancements address this issue.

We will briefly overview previous TOS version, emphasizing the motivation and integration of the enhancements. TOS data contracts are presented in full detail in deliverable D.3.5 First Revision of the WP3 framework.

Terminological and ontological resources are encoded into data structures named coded concept. Concepts are grouped and indexed into data structures such as Code Systems, Vocabulary Domain and Value Sets. **Figure 3** presents the relationships between the main components of TOS data. The **code system** contains a

D.3.3 First Revision Data & Process model system framework

list of coded concepts from a particular domain of discourse. A **vocabulary domain** serves as the link between an HL7 coded attribute and the set(s) of valid concept codes for that attribute. A vocabulary domain represents an abstract conceptual space from which the values of an attribute can be drawn - such as "countries of the world", "the gender of a person used for administrative purposes", etc. Before an attribute can be used in a message, however, the actual list of concept codes needs to be defined. A list of valid concept codes is referred to as a **value set**.

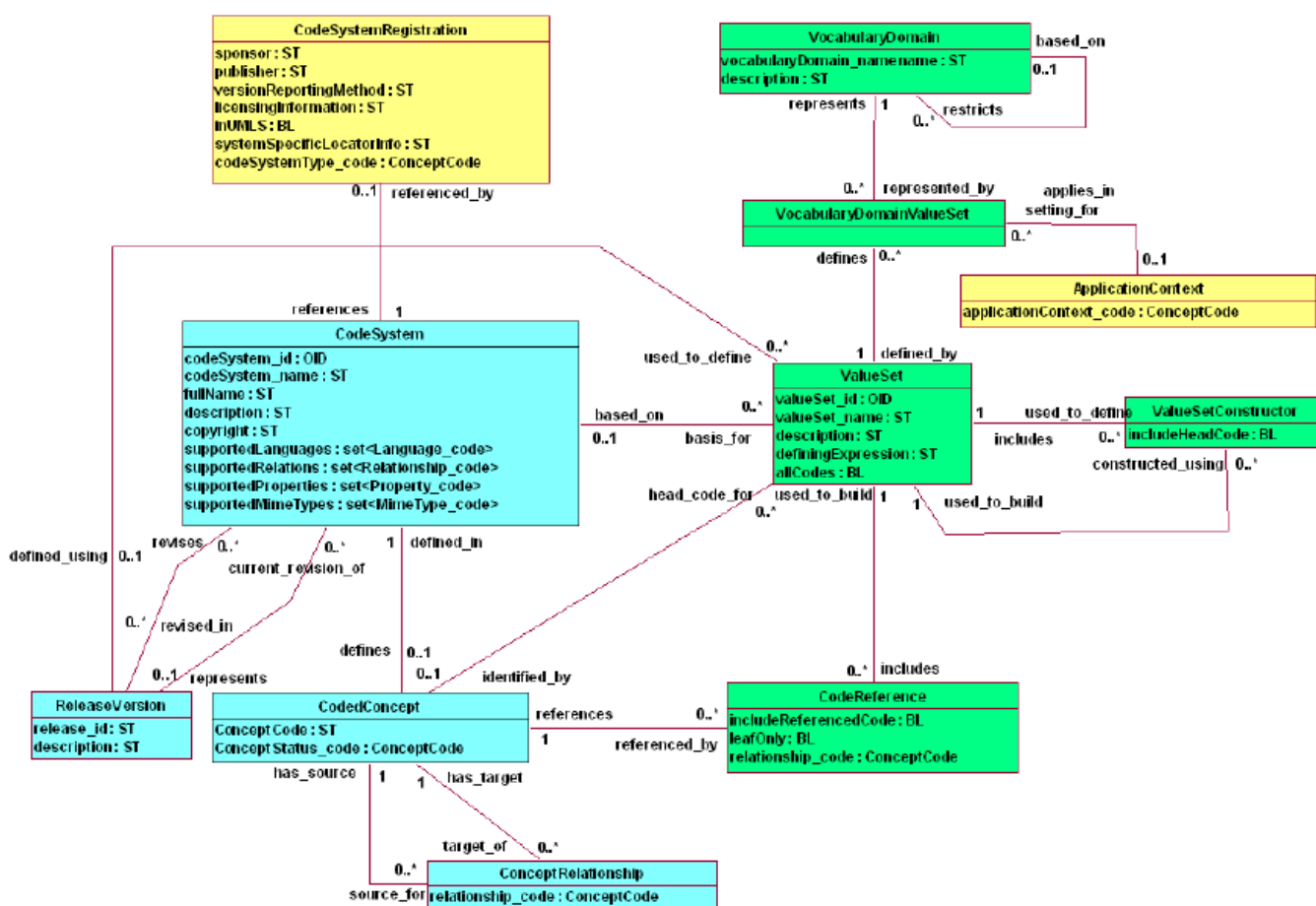


Figure 3 – Relationships between the main components of TOS data

A detailed overview of Code Systems illustrates both concepts and concept properties, along with information about how concept properties are actually defined at code system level (see **Figure 4**).

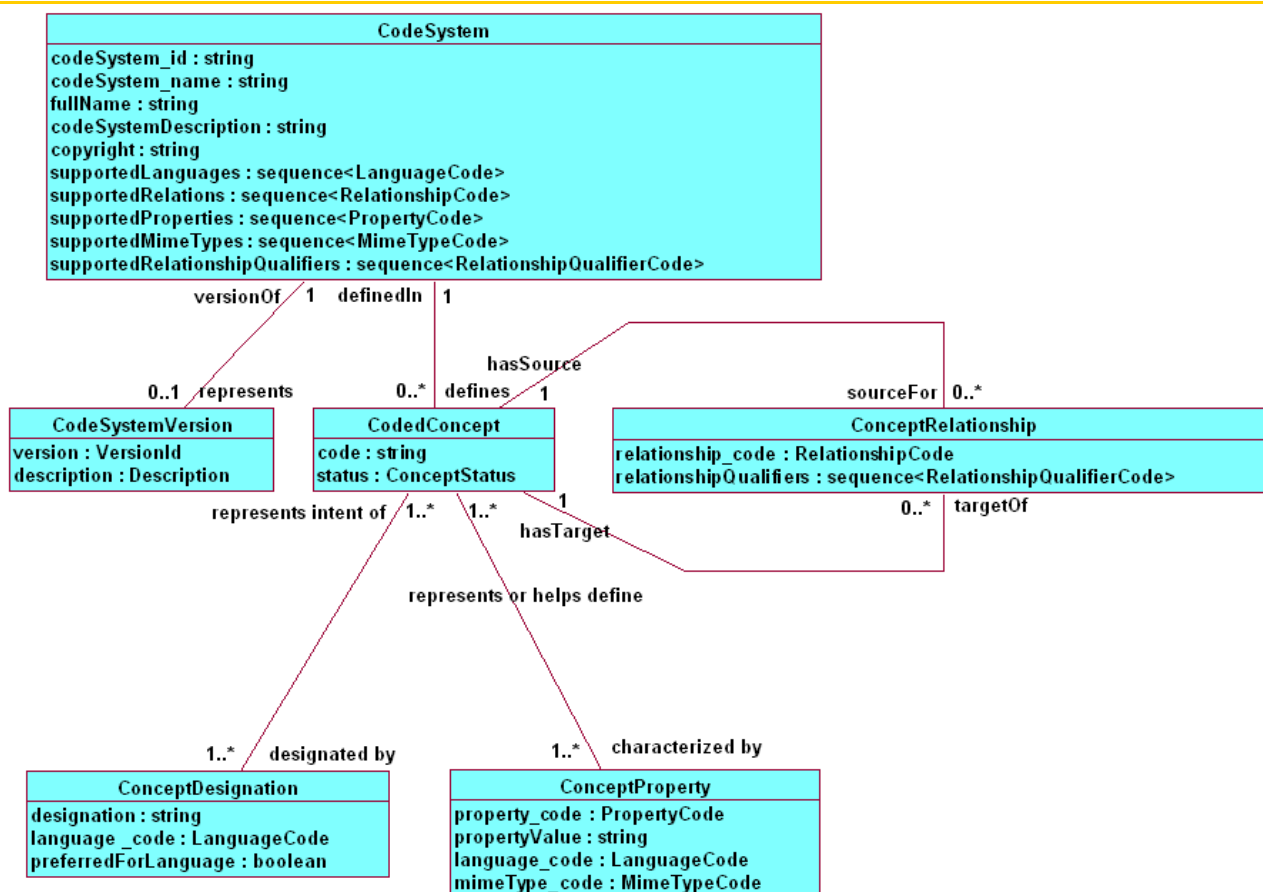


Figure 4 – Code systems representation illustrating concepts and concept properties

Relevant to TOS enhancements are the following attributes - as they carry additional metadata about a code system:

- *supportedLanguages* - The list of all of the languages that are fully or partially supported by the code system. A "supported" language is recognized by the code system and at least some of the concept designations or properties are available in that language. All code systems must support at least one language. While the service must list all of the primary language subtags that it supports, it is optional whether it lists secondary languages. (i.e., if it supports "en-UK", it must list "en" but may or may not list "en-UK").
- *supportedProperties* - The property codes supported by the code system. Whenever possible, however, property codes should use the HL7 ConceptProperty code system (OID 2.16.840.1.113883.5.1087).
- *supportedMimeType* - A list of the MIME types used in designations, descriptions or properties in the code system. These codes must be drawn from the officially designated HL7 Media Type code system, which is currently OID 2.16.840.1.113883.6.79 - MediaType. The text/plain MIME type must be supported by all code systems.

Having in mind how TOS understands to structure its data regarding concepts and their associated properties, we present in **Figure 5** the actual data contract that a TOS Consumer will use in the additional operations defined on Message Layer Browsing.

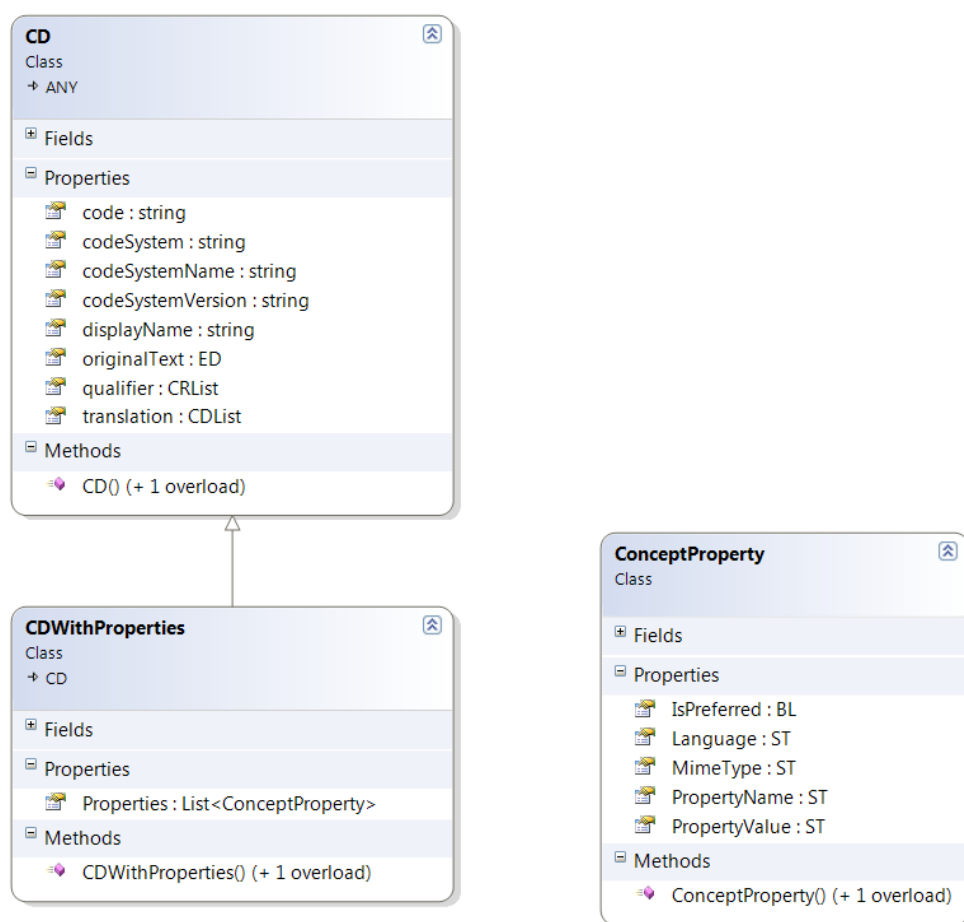


Figure 5 – Data Contract for the additional TOS operations

The data type *CDWithProperties* represents a concept descriptor with properties, extending the *CD* data type (the simpler concept descriptor used in previous TOS version). A concept property includes information about the name of the property, the value in the language specified, with corresponding mime type and *IsPreferred* indicates if the property value is preferred for the language.

The basic HL7 Data Types BL and ST that appear in the *ConceptProperty* class are reminded below:

BL	Boolean	<i>BL</i> stands for the values of two-valued logic. A <i>BL</i> value can be either <i>true</i> or <i>false</i> , or may be NULL.
ST	Character string	The character string data type stands for text data, primarily intended for machine processing (e.g., sorting, querying, indexing, etc.) Used for names,

		symbols, and formal expressions.
--	--	----------------------------------

4.1.5. Fault contracts

Class	Description
BadlyFormedMatchText	The supplied match text was syntactically incorrect for the specified match algorithm
NoApplicableValueSet	The service was unable to determine which value set should be used for vocabularyDomain_name and applicationContext_code.
TimeoutError	The time limit specified for the function call has expired
UnexpectedError	An error that could not be anticipated by the specification has occurred. This includes things like memory faults, I/O errors, database access errors and any other sort of unexpected event that prevents successful completion of the API call. possible_cause can carry a more detailed description of what actually occurred.
UnknownLanguageCode	language_code isn't supported by the code system
UnknownMatchAlgorithm	The supplied match algorithm isn't supported by the service
UnknownValueSet	<ol style="list-style-type: none"> 1. A value set id was supplied, but it isn't recognized by the service or 2. Only a value set name was supplied the name isn't recognized by the service 3. Neither a name or an id was supplied
UnknownVocabularyDomain	The vocabularyDomain_name isn't recognized by the service

4.1.6. Database layer

The database layer of the TOS is made of a complex table structure, from which a selection containing only tables directly related to concept and concept properties modeling is presented in **Figure 6**. We will present existing and added implementation related to the retrieval of concepts with properties.

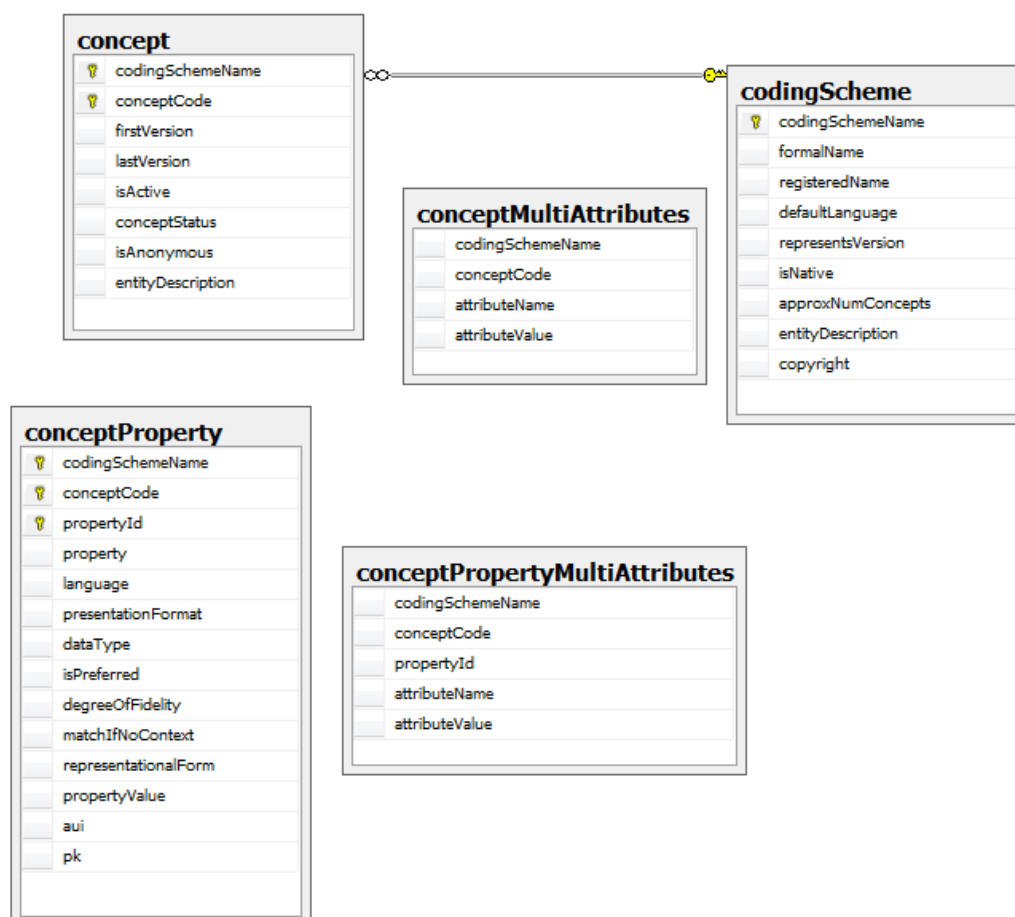


Figure 6 – Coding schema database structure - selection regarding concepts and concept properties

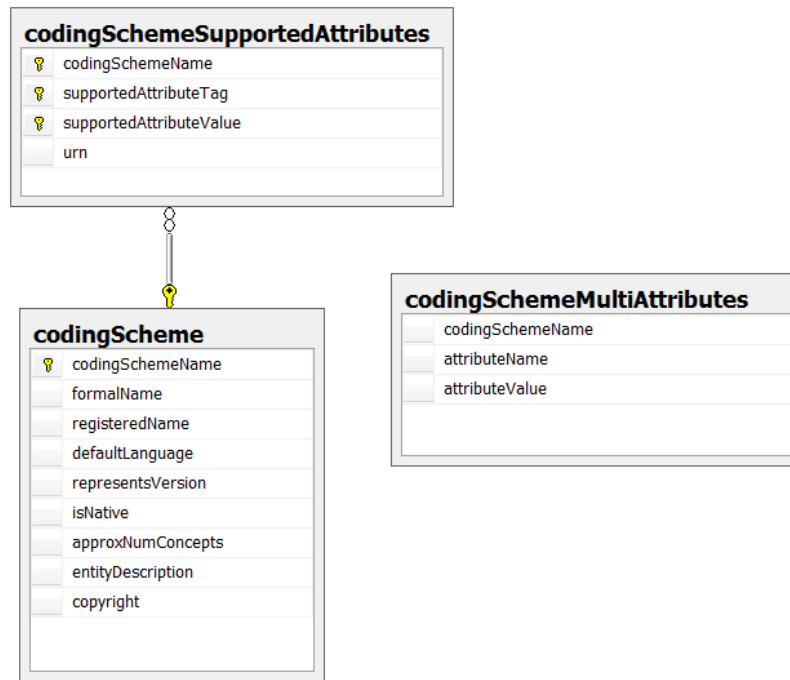


Figure 7 – Coding schema database structure - selection regarding the coding scheme and its attributes

The coding schema database structure is based on LexGrid documentation for HL7 CTS1. The database structure follows the UML diagram for coding schemas presented below:

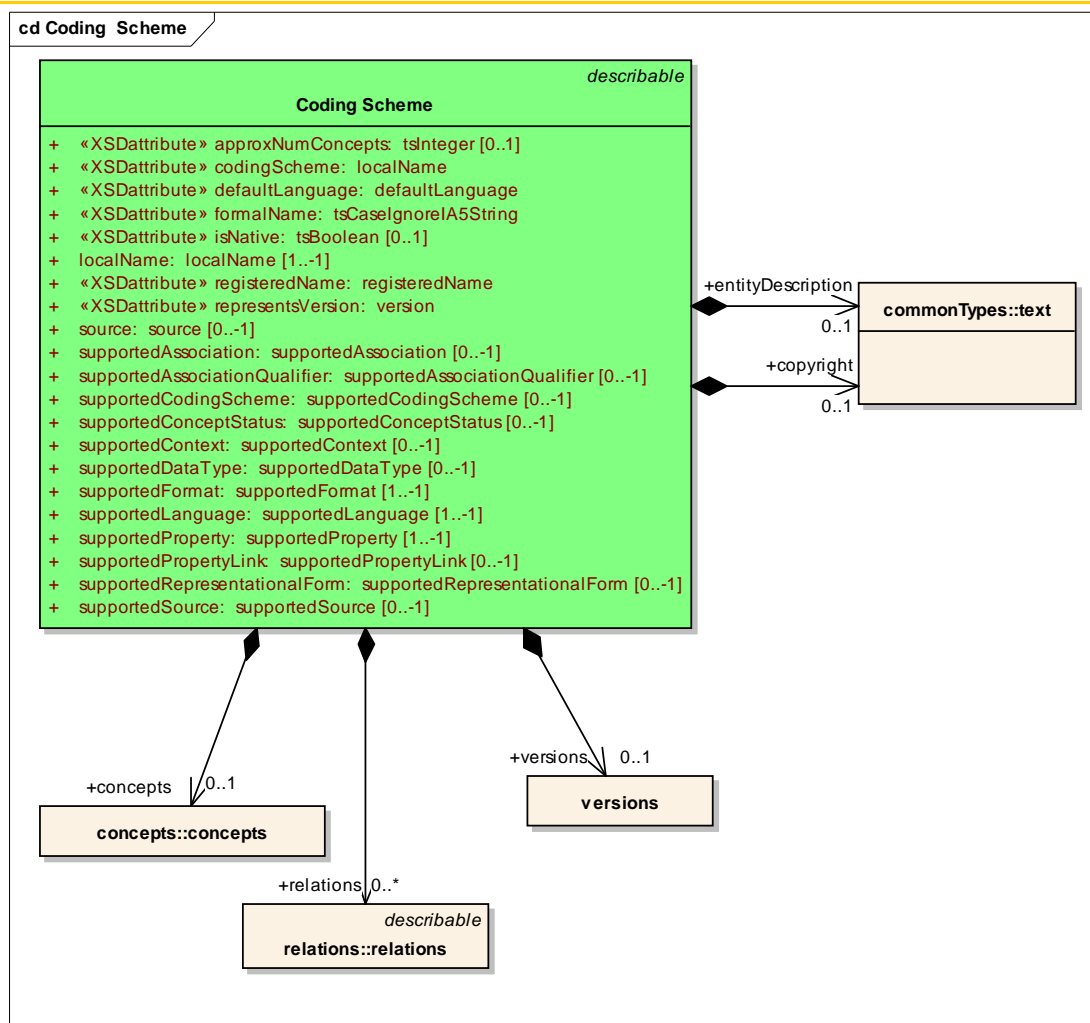


Figure 8 – Coding schema class diagram

As it can be seen from **Figure 8**, supported properties for concepts are defined at the coding scheme level. All concepts in the coding scheme must obey the definition of the coding scheme they belong to.

The implementation of the concept properties into the database can be further explained by returning to **Figure 6**. The concept properties defined at coding scheme level are a kind of *supportedAttributes* of the coding scheme. From the relational point of view, the associated concept properties for the scheme are inserted in the related table **codingSchemeSupportedAttributes**, along with other schema level attributes. The actual concretization of the property for a certain concept finds its place in the **conceptProperty** table.

For extending the API for the retrieval of concepts with properties, the development steps involved were as following:

- Defining operations on the service contract (as functionally detailed in the corresponding section – see **4.1.3**)
- Adding SQL procedures for retrieving the information from the database
- Adding Data Sets to store the parameters for and the results from the SQL procedures.

The enhancements started with defining SQL procedures that realize the additional lookup operations that were requested on the service contract. Each lookup operation had its corresponding SQL procedure. We defined also additional datasets for both input parameters and results of each new procedure.

At run-time, the TOS service receives and extracts actual parameters from the received web service queries, fills the datasets with parameters, requests the execution of the SQL procedures on the TOS database repository and then retrieves the resulting dataset with matching concepts. After internal processing, a message is constructed according to the defined data contract and is returned to the TOS consumer.

4.2. Master Patient Index Service

4.2.1. Architectural overview

The Master Patient Index (MPI) Service serves as a Web Service responsible for patient management within the ReMINE project. Based on the HSSP EIS (Entity Identification Service – see specifications in [1]), it is responsible for CRUD (Create, Read, Update, and Delete) scenario of patients within the metadatabase services. Patients will be stored using entity-identifying characteristics (e.g. demographic information) and manipulated through a patient id provided by MPI after creation of patients. HSSP EIS specifications are for the management of medical entities (patients, doctors, medical organizations, clinics, hospitals, etc.). MPI relies on the HSSP EIS specifications for patient management. The below figure explains a common MPI deployment within an organization.

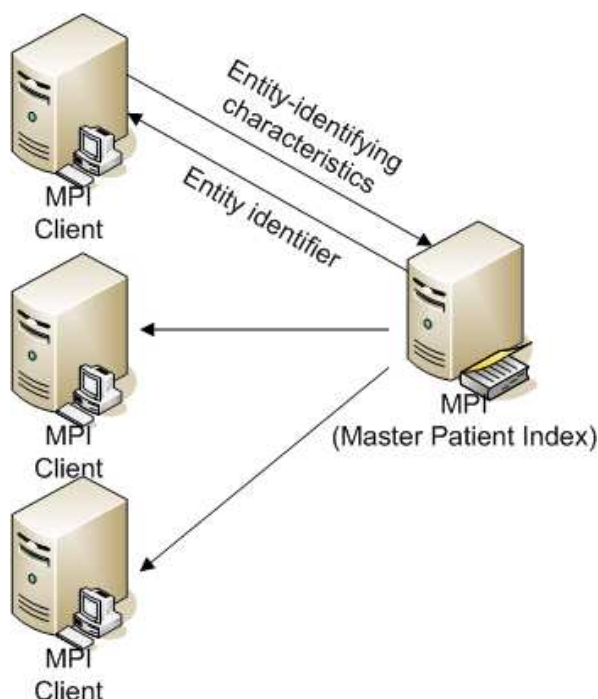


Figure 9 – Representative deployment of MPI within an organization

4.2.2. Business scenarios and use cases

The classes of actors that are anticipated to be users of the MPI include:

- **MPI software consumers** – software applications that use MPI within ReMINE project (the Correlator activated through DEMS)
- **Software Developers** – The people who build the software that creates, validates and processes HL7 Version 3 messages.

Patient Creation – The following trigger mechanism is enabled: HIS -> H-ERP Bridge -> DEMS -> Correlator -> MPI when a patient is to be created. First, MPI will be queried to see if the patient exists, if not the patient will be created.

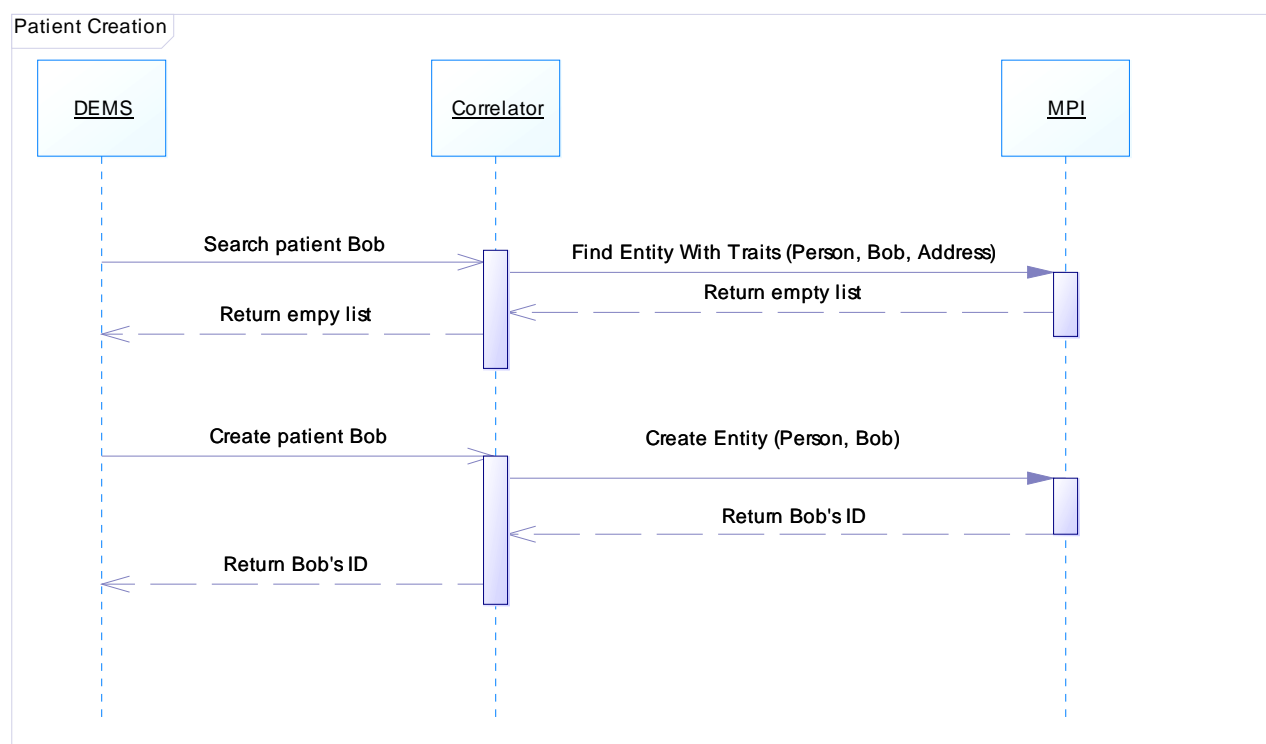


Figure 10 – Patient creation scenario

Patient Update – It usually refers to updating patient details, done in the same automatically manner as patient creation.

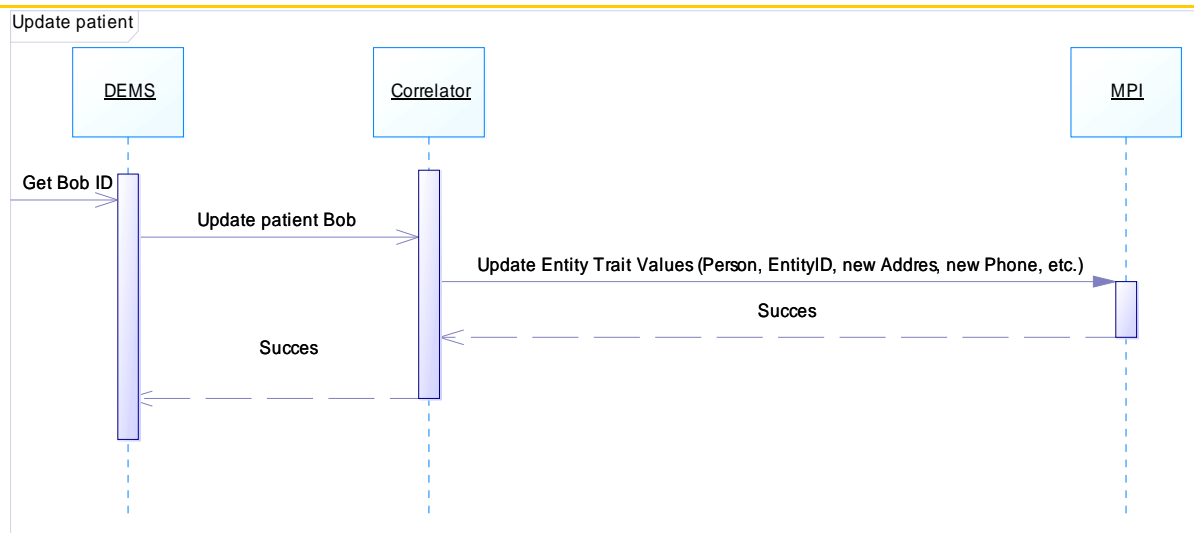


Figure 11 – Patient update demographics scenario

Patient delete– It usually refers to updating patient details, done in the same automatically manner as patient creation. MPI provides logical deletion (Inactivate entity) and the complementary function activate entity (restore a logical deleted patient).

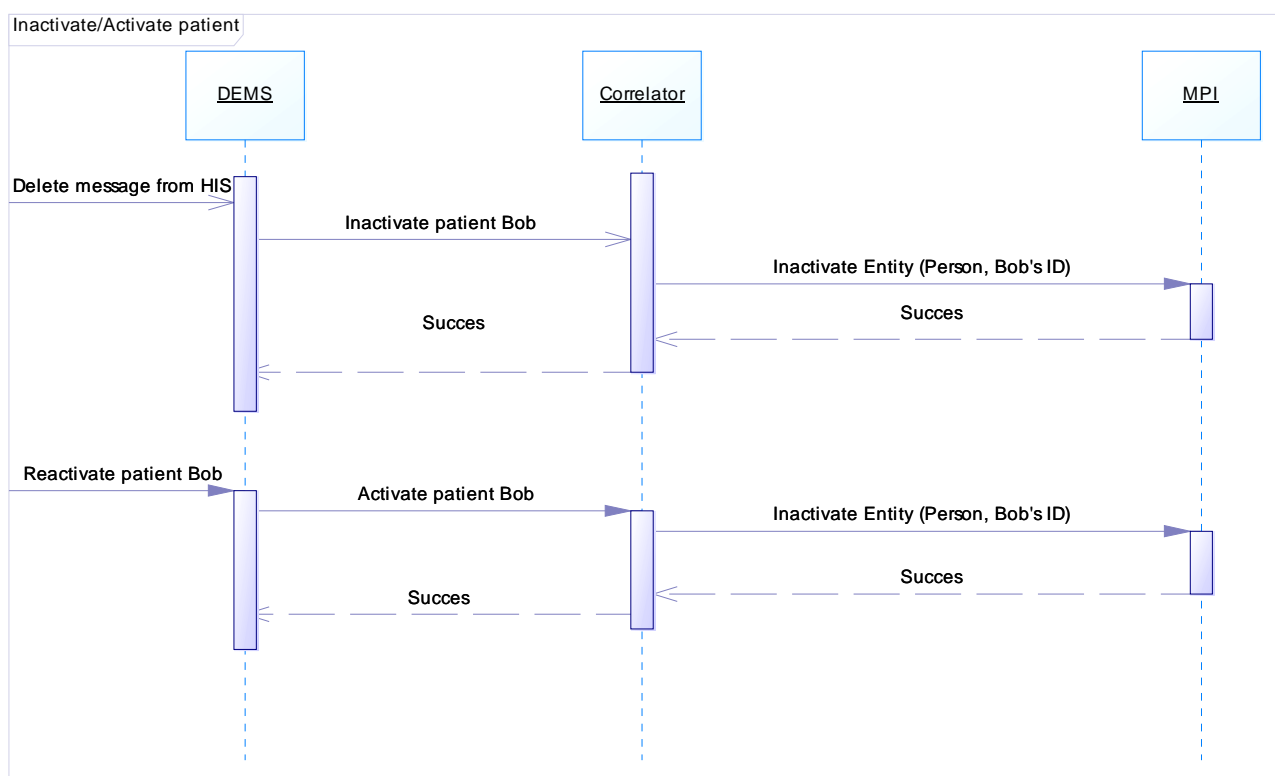
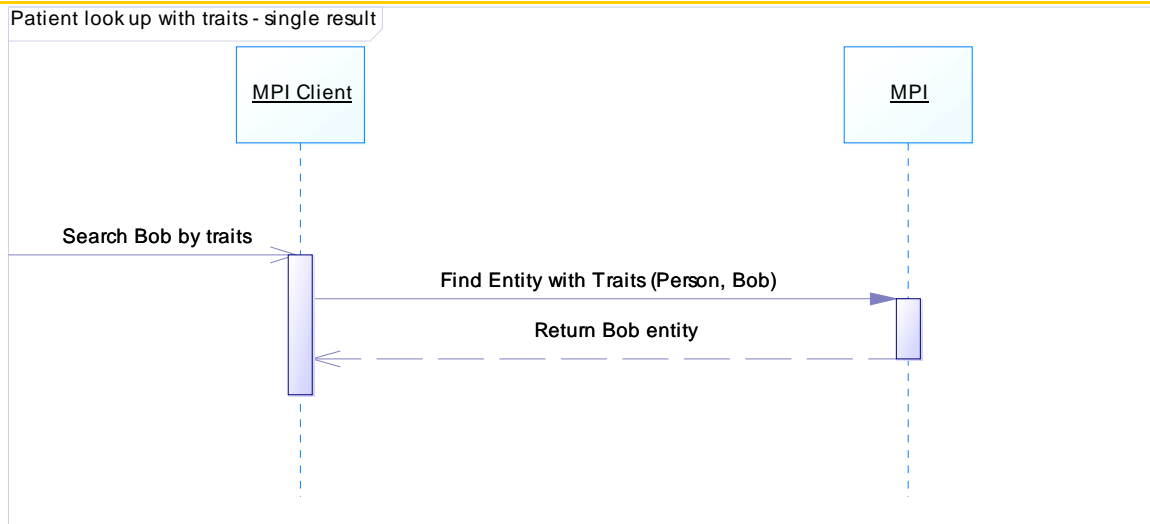
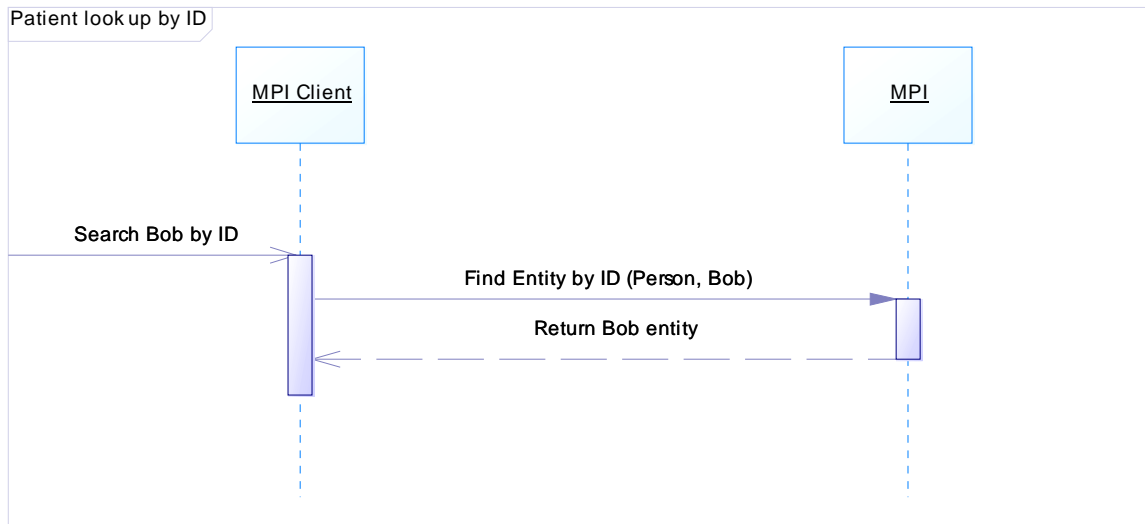


Figure 12 – Patient deletion/activation

Physical removal is done in the same manner using the Purge Entity method on MPI.

Patient search – Patient search (query) can be done automatically (through DEMS-> Correlator) or even manual by all liveware entities (doctors, nurses, risk manager).

**Figure 13** – Patient search based on input traits – with one result**Figure 14** – Patient search based patient ID – returns one ore zero entities

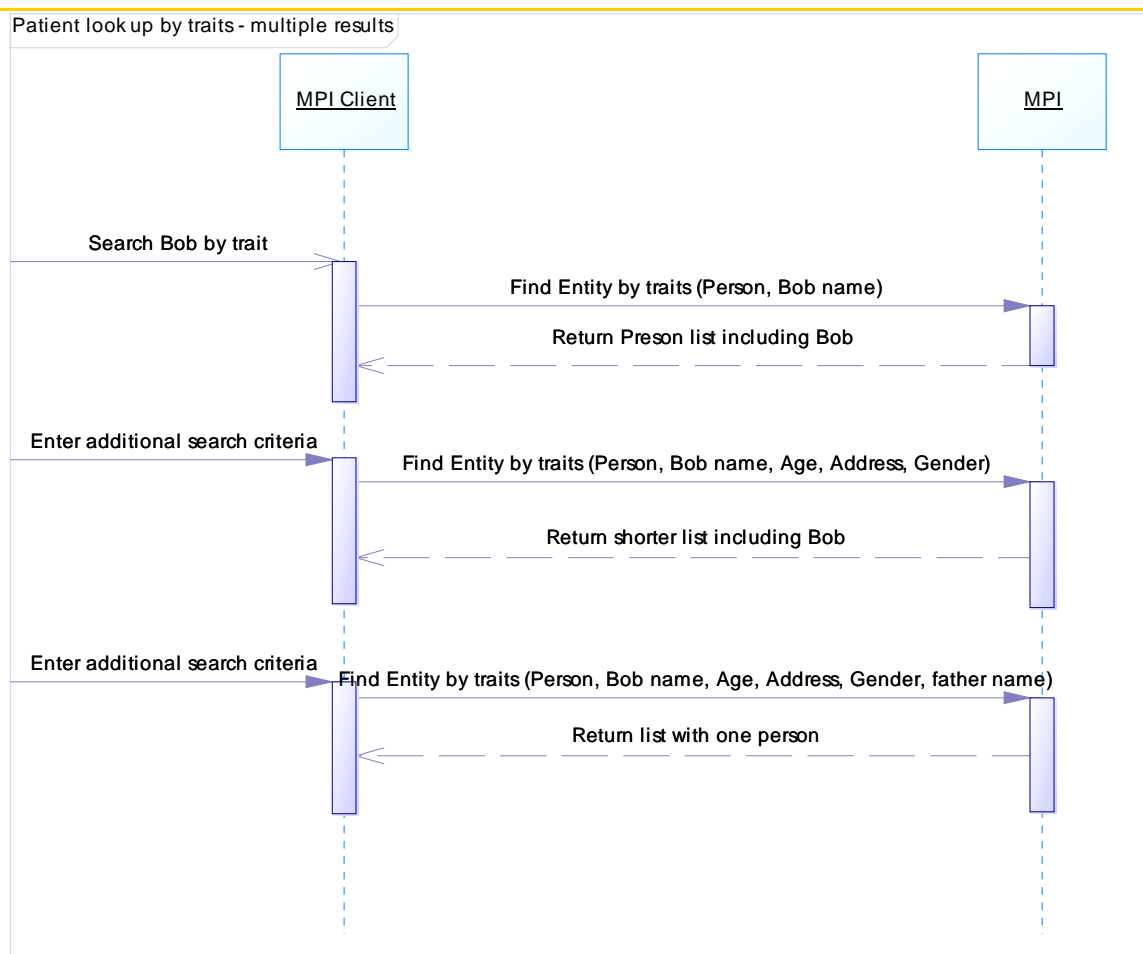


Figure 15 – Patient search based on input traits – with multiple results

4.2.3. Service contracts

The detailed design model consists of the three sections: service contracts, data contracts, fault contracts as is defined in .NET WCF (Windows Communication Foundation). In WCF, all services expose contracts. The contract is a platform-neutral and standard way of describing what the service does.

- **Service contracts** describe which operations you can perform on the service.
- **Data contracts** define which data types are passed to and from the service. WCF defines implicit contracts for built-in types such as int and string, but you can easily define explicit opt-in data contracts for custom types.
- **Fault contracts** define which errors are raised by the service, and how the service handles and propagates errors to its clients.

MPI exposes several API interfaces the relevant ones are these:

- **IEntityManagement** - Entity Management provides operations for manipulation of Entity Identifiers and traits. Used for to create/update/delete a patient.
- **IQueryFunctions** - Query Functions provides query operations for discovering entity identifiers and traits. Used to look up patients based on the above described use cases.
- **IAdministration** - Administrative Functions: These functions are expected to be used only by systems administrators who maintain the operations of EIS instances. Used for physical deletion of entities
- **IServiceMetadataManagement** – This set of functions in this set includes the management of traits, i.e. creation, retrieval, updating and deletion of traits.

4.2.4. Detailed functional model on each interface

Entity management layer (IEntityManagement)

- ActivateEntity

Business friendly name	Activate an inactive patient
Signature	<pre>void ActivateEntity(string domainIdentifier, string entityTypeIdentifier, string entityIdIdentifier);</pre>
Description	Changes an Inactive Entity to an Active state
Precondition	<ul style="list-style-type: none"> • There is a patient (Person Entity) that has an inactive state
Inputs	<p>domainIdentifier = Entity Domain Identifier, current active domain: ReMINE</p> <p>entityTypeIdentifier = Entity Type Identifier, type of entity: "HL7-RIM-V3-Person"</p> <p>entityIdentifier = Entity Identifier, patient ID</p>
Outputs	Confirmation that the Entity identified by the Identifier is now active.
Post conditions	Entity Status is active
Exception Conditions	<ul style="list-style-type: none"> • Identifier is not recognized as identifying an entity (EntityTypeIdentifierDoesNotExist)

(with fault contracts)	<ul style="list-style-type: none"> Identifier identifies an entity that is active (EntityIsActive) Entity identifier is not assigned to an entity type on domain (EntityIdentifierNotAssignedToEntityTypeOnDomain) Entity type is not assigned to a domain (EntityTypeNotAssignedToDomain) Domain identifier is not known to the system (DomainIdentifierDoesNotExist) Other unexpected error (UnexpectedError)
-------------------------------	--

- CreateEntity

Business friendly name	Creates a patient in MPI
Signature	<pre>string CreateEntity(Entity entity);</pre>
Description	Allows for creation of an entity with a list of trait values.
Precondition	Entity domain and type exist in MPI.
Inputs	Patient entity
Outputs	New patient id.
Post conditions	A new patient with specific traits has been created.
Exception Conditions (with fault contracts)	<ul style="list-style-type: none"> A value is null Domain does not exist on system Entity Type does not exist on system Entity type not assigned to domain Duplicates exists in Entity TraitList Values of all mandatory traits for the Entity Type have not been provided A trait in the list does not exist in the system metadata for the Entity Type

	<ul style="list-style-type: none"> • A trait in the trait list is not assigned to entity type on domain • Data type of a trait is unknown
--	---

- InactivateEntity

Business friendly name	Logical deletion of a patient
Signature	<pre>void InactivateEntity(string domainIdentifier, string entityTypeIdentifier, string entityIdentifier);</pre>
Description	Makes an Entity inactive, i.e. will not appear in search result sets.
Precondition	The Identifier identifies an Entity that is active
Inputs	<p>domainIdentifier = Entity Domain Identifier, current active domain: ReMINE</p> <p>entityTypeIdentifier = Entity Type Identifier, type of entity: "HL7-RIM-V3-Person"</p> <p>entityIdentifier = Entity Identifier, patient ID</p>
Outputs	Confirmation that the Entity identified by the Identifier is now inactive
Post conditions	Entity status is set to inactive
Exception Conditions (with fault contracts)	<ul style="list-style-type: none"> • Domain does not exist on system • Entity Type does not exist on system • Entity type not assigned to domain • Identifier is not recognized as identifying an entity • Entity identifier not assigned to entity type on domain • Identifier identifies an Entity that is inactive

- UpdateTraitValues

Business	Update patient demographics.
-----------------	------------------------------

friendly name	
Signature	<code>void UpdateTraitValues(Entity entity);</code>
Description	Allows for addition and/or update of a set of trait values for an entity specified by a unique Entity Identifier.
Precondition	The Entity specified by the Entity Identifier exists.
Inputs	Entity with specified traitlist.
Outputs	An acknowledgement that the trait values have been successfully added and/or updated
Post conditions	Entity updated with trait values
Exception Conditions (with fault contracts)	<ul style="list-style-type: none"> • A value is null • Domain does not exist on system • Entity Type does not exist on system • Entity type not assigned to domain • Identifier is not recognized as identifying an entity • Entity identifier not assigned to entity type on domain • Duplicates exists in Entity TraitList • Values of all mandatory traits for the Entity Type have not been provided • A trait in the list does not exist in the system metadata for the Entity Type • A trait in the trait list is not assigned to entity type on domain • Data type of a trait is unknown

- PurgeEntity

Business friendly name	Physically remove a patient
-------------------------------	-----------------------------

Signature	<pre>void PurgeEntity(string domainIdentifier, string entityTypeIdentifier, string entityIdentifier);</pre>
Description	Allows for the “complete” removal of an Inactive Entity from an EIS service.
Precondition	The Entity should exist and should be in an Inactive State. All other roles to this entity must be removed first.
Inputs	<p>domainIdentifier = Entity Domain Identifier, current active domain: ReMINE</p> <p>entityTypeIdentifier = Entity Type Identifier, type of entity: "HL7-RIM-V3-Person"</p> <p>entityIdentifier = Entity Identifier, patient ID</p>
Outputs	An acknowledgement that the Entity has been removed
Post conditions	None
Exception Conditions (with fault contracts)	<ul style="list-style-type: none"> Identifier is not recognized as identifying an entity Identifier identifies an entity that is active Domain does not exist on system Entity Type does not exist on system Entity type not assigned to domain Entity identifier not assigned to entity type on domain

- FindEntitiesByTrait

Business friendly name	Look up patients based on input traits (e.g. name, surname, gender, etc.)
Signature	<pre>List<Entity> FindEntitiesByTrait(TraitList TraitListInput, List<string> TraitListOutputIdentifiers, List<string> DomainIdentifiers, List<string> EntityTypeIdentifiers);</pre>

Description	Given a list of Traits (Trait Identifier , Trait Value, Matching Algorithm), this allows for a search of matching Entities.
Precondition	The Traits specified in the list exist in the system
Inputs	<ul style="list-style-type: none"> • Input Trait list: A list of (Trait Name or Trait Identifier, Trait Value) pairs • TraitListOutputIdentifiers – a list of output trait identifiers requested • DomainIdentifiers – list of domains to search. Only “ReMINE” • EntityTypeIdentifiers – list of entity types to query. Only “HL7-RIM-V3-Person” for patient
Outputs	A list of entities found based on the query, or an empty list.
Post conditions	None
Exception Conditions (with fault contracts)	<ul style="list-style-type: none"> • A value is null • Domain does not exist on system • Entity Type does not exist on system • Entity type not assigned to domain • Identifier is not recognized as identifying an entity • Entity identifier not assigned to entity type on domain • A matching algorithm for a trait is not supported • A Trait Identifier in the input list does not exist • A trait in the list is not assigned to entity type on domain • A trait in the input list has a invalid data type • A trait in the input list has a invalid value • A trait in the input list is not assigned to domain • A trait in the input list is not assigned to entity type

- GetInformationForEntity

Business friendly name	Retrieve a patient by ID.
Signature	<pre>Entity GetInformationForEntity(string domainIdentifier, string entityTypeIdentifier, string entityIdentifier);</pre>
Description	Retrieves all information for an Entity known by the MPI(trait, status etc)
Precondition	The Identifier identifies an Entity that exists
Inputs	<p>domainIdentifier = Entity Domain Identifier, current active domain: ReMINE</p> <p>entityTypeIdentifier = Entity Type Identifier, type of entity: "HL7-RIM-V3-Person"</p> <p>entityIdentifier = Entity Identifier, patient ID</p>
Outputs	<p>Status and all trait values of the Entity identified by the Identifier</p> <p>If entity has been merged into another "main" identifier, then return active identifier.</p>
Post conditions	None
Exception Conditions (with fault contracts)	<ul style="list-style-type: none"> • A parameter value is null • Domain does not exist on system • Entity Type does not exist on system • Entity type not assigned to domain • Identifier is not recognized as identifying an entity • Entity identifier not assigned to entity type on domain

4.2.5. Data contracts

Data contracts define which data types are passed to and from the service. WCF defines implicit contracts for built-in types such as int and string, but you can easily define explicit opt-in data contracts for custom types. MPI relies on HSSP EIS specifications and the data structure is based on HL7 RIM model.

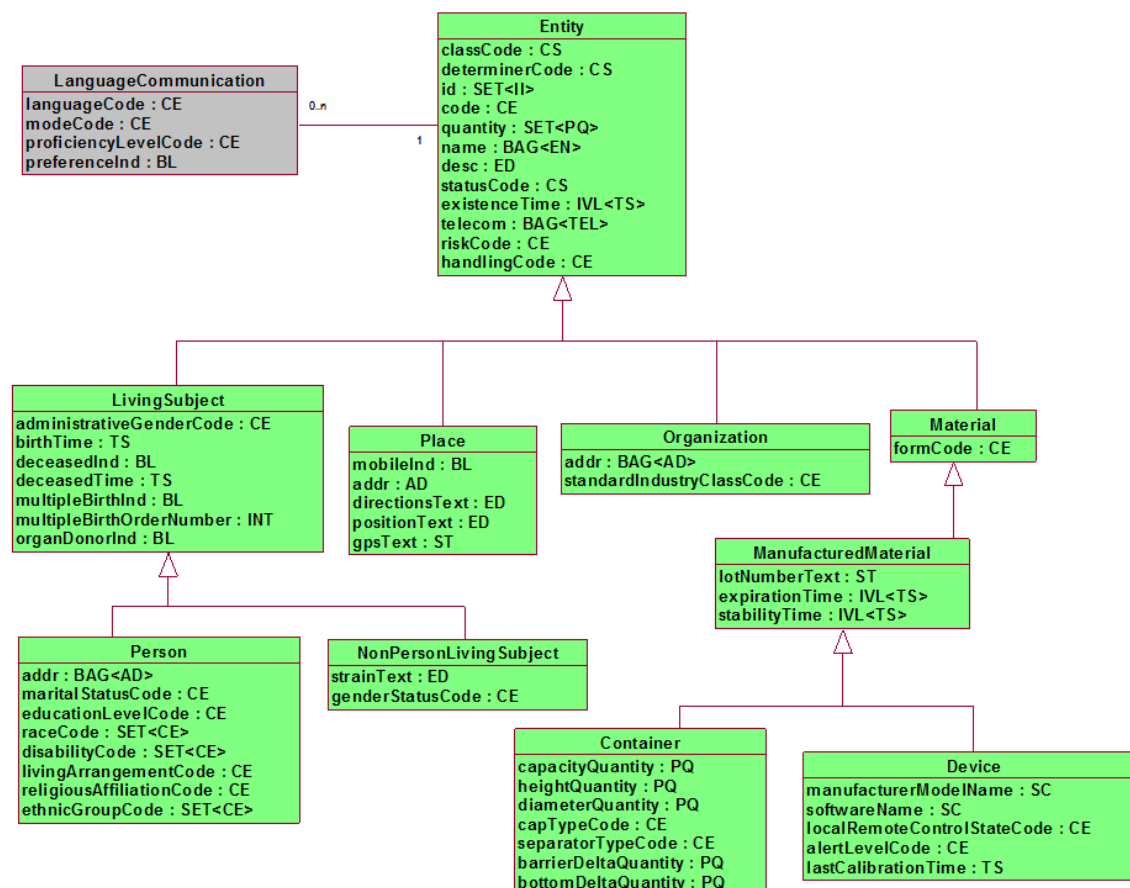


Figure 16 – HL7 Reference Information Model (RIM)

As can be seen from the entity specification from the HL7 RIM, entities are uniquely defined with distinct attributes based on subclass. This entity uniqueness must be supported by the MPI service interface. This will be achieved through the definition of Entity Types, as described above and then by defining standard traits that are mandatory or optional for each type. Since each entity type has potentially different identifying and non-identifying attributes, it is important to be able to explicitly state what they are for both the service consumer and the service provider.

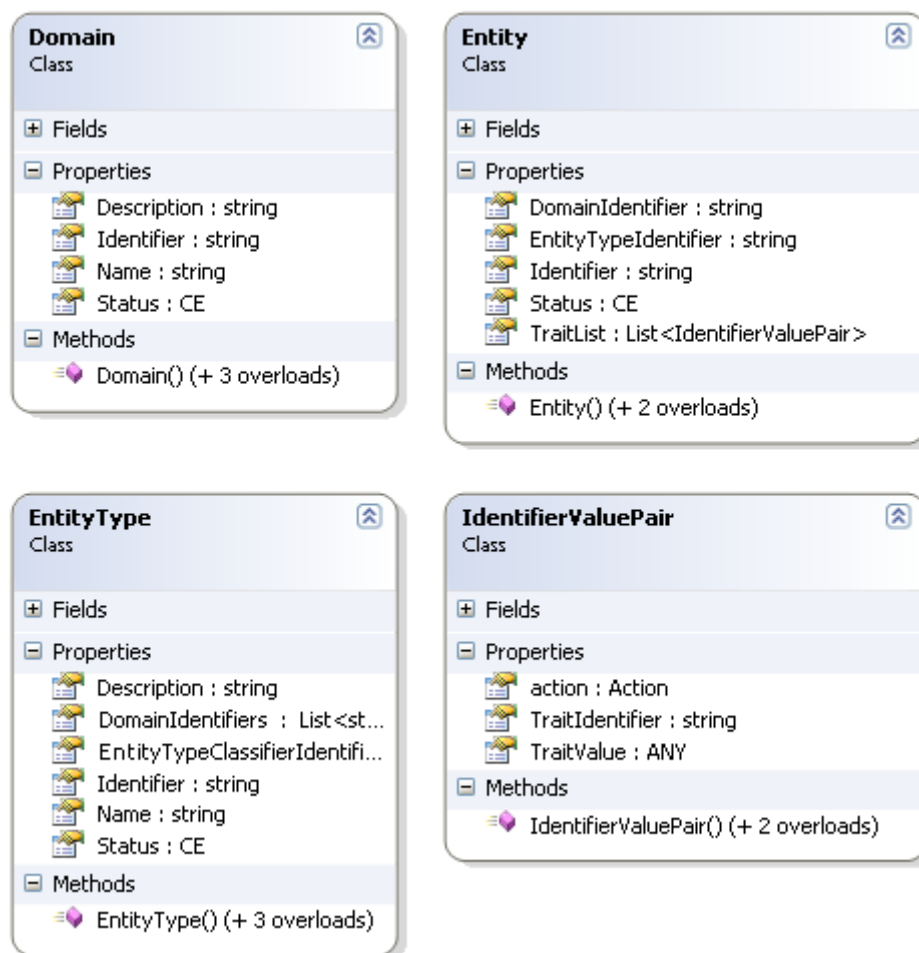


Figure 17 – Basic MPI classes

Class	Description
Domain	There is only one ReMINE identifier for domain: "ReMINE" . Status can be: active, terminated, nullified. CE is a HL7 data type.
EntityType	Represents types of entities. For patient there will be only one: "HL7-RIM-V3-Person" for EntityType identifier . Other possible entity types are based on HL7 RIM.
Entity	<ul style="list-style-type: none"> - identifier is generated by the MPI service at CreateEntity method - status: active, terminated, nullified - domainIdentifier: to which domain it belongs

D.3.3 First Revision Data & Process model system framework

	<ul style="list-style-type: none"> - EntityTypeIdIdentifier: type of entity. Only "HL7-RIM-V3-Person" - TraitList: a list of IdentifierValuePair where the traits are stored. E.g. Patient sex is stored as ADMINISTRATIVE_GENDER as TraitIdentifier and Traitvalue is a CE (coded concept, e.g.: <code="M" displayName="Male" codeSystem="AdministrativeGender")
IdentifierValuePair	<ul style="list-style-type: none"> - Identifies a Trait. Has TraitIdentifier (name of trait) and TraitValue – value of Trait (derives from HL7 ANY data tupe)

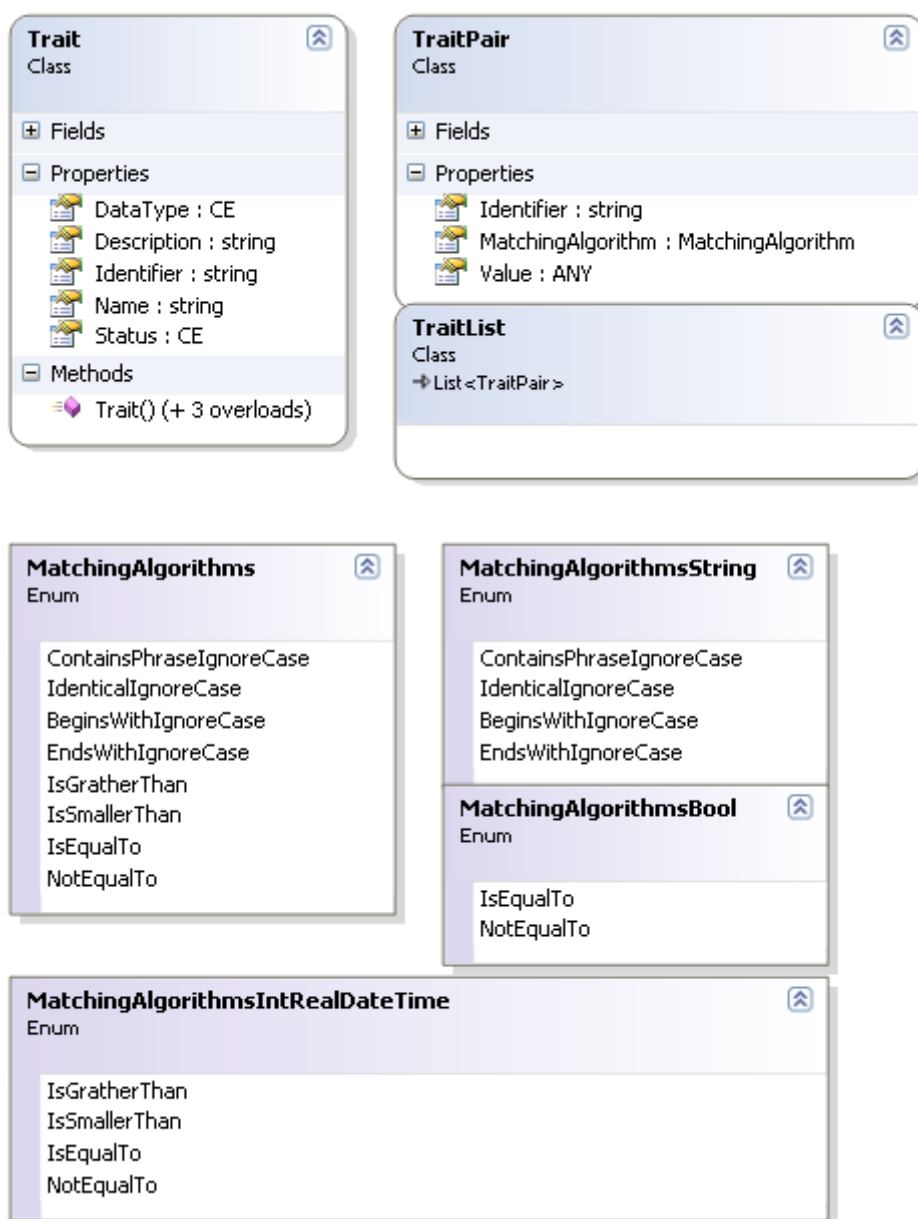


Figure 18 – MPI query classes

The above figure displays classes used to build the parameters to query MPI in FindEntitiesByTrait.

Class	Description
TraitPair	Class for query traits. Matching algorithm is one of the matching algorithms used for the query. Example: MatchingAlgorithmsString.BeginsWithIgnoreCase – trait is a string and it will be matched on MPI ignoring case and beginning with text to match.
Trait	Class for definition of a trait metadata. <ul style="list-style-type: none"> - DataType: Data type of Trait - Description: Trait description - Identifier: Trait identifier - Name: name of trait - Status: trait status(active, inactive)

The description of the common HL7 Data Types is given below:

Data type	Name	Description
AD	Postal Address	Mailing and home or office addresses. A sequence of address parts, such as street or post office Box, city, postal code, country, etc.
ANY	Anything	Defines the basic properties of every data value. This is an abstract type, meaning that no value can be just a data value without belonging to any concrete type. Every concrete type is a specialization of this general abstract DataValue type.
CD	Concept description	A concept descriptor represents any kind of concept usually by giving a code defined in a code system. A concept descriptor can contain the original text or phrase that served as the basis of the coding and one or more translations into different coding systems. A concept descriptor can also contain qualifiers to describe, e.g., the concept of a "left foot" as a postcoordinated term built from the primary code "FOOT" and the qualifier "LEFT". In cases of an exceptional value, the concept descriptor

		need not contain a code but only the original text describing that concept.
CE	Coded with Equivalents	Coded data that consists of a coded value (CV) and, optionally, coded value(s) from other coding systems that identify the same concept. Used when alternative codes may exist.
CS	Simple coded values	Coded data in its simplest form, where only the code is not predetermined. The code system and code system version are fixed by the context in which the CS value occurs. CS is used for coded attributes that have a single HL7-defined value set.
CV	Coded Value	Coded data, specifying only a code, code system, and optionally display name and original text. Used only as the data type for other data types' properties.
ED	Encapsulated data	Data that is primarily intended for human interpretation or for further machine processing outside the scope of HL7. This includes unformatted or formatted written language, multimedia data, or structured information in as defined by a different standard (e.g., XML-signatures.)
EN	Entity Name	A name for a person, organization, place or thing. A sequence of name parts, such as given name or family name, prefix, suffix, etc
II	Instance identifier	An identifier that uniquely identifies a thing or object. Examples are object identifier for HL7 RIM objects, medical record number, order id, service catalog item id, Vehicle Identification Number (VIN), etc. Instance identifiers are defined based on ISO object identifiers.
IVL	Interval	A set of consecutive values of an ordered base data type. Any ordered type can be the basis of an interval; it does not matter whether the base type is discrete or continuous. If the base data type is only partially ordered, all elements of the interval must be elements of a totally ordered subset of the partially ordered data type.
ON	Organization Name	An EN used when the named Entity is an Organization. A sequence of name parts.
PN	Person Name	An EN used when the named Entity is a Person. A sequence of name parts, such as given name or family name, prefix, suffix, etc. A name part is a restriction of entity name part that only allows those entity name parts qualifiers applicable to person names. Since the structure of entity name is mostly determined by the requirements of person name, the restriction is very minor.

D.3.3 First Revision Data & Process model system framework

PQ	Physical Quantity	A dimensioned quantity expressing the result of measuring.
RTO	Property of ratio	The quantity that is being divided in the ratio. The default is the integer number 1 (one.)
SC	Character string with code	A character string that optionally may have a code attached. The text must always be present if a code is present. The code is often a local code.
ST	Character string	The character string data type stands for text data, primarily intended for machine processing (e.g., sorting, querying, indexing, etc.) Used for names, symbols, and formal expressions.
TEL	Telecommunication address	A telephone number (voice or fax), e-mail address, or other locator for a resource mediated by telecommunication equipment. The address is specified as a Universal Resource Locator (URL) qualified by time specification and use codes that help in deciding which address to use for a given time and purpose.
TS	Timestamp	A quantity specifying a point on the axis of natural time. A point in time is most often represented as a calendar expression. Semantically, however, time is independent from calendars and best described by its relationship to elapsed time (measured as a physical quantity in the dimension of time.) A point in time plus an elapsed time yields another point in time. Inversely, a point in time minus another point in time yields an elapsed time.

The current traits for **patient entity** are:

Trait Identifier	Trait Data Type	Description
CODE	CS	A value representing the specific kind of Entity the instance represents. No duplicates allowed in the entity traits list.
DESCRIPTION	ED	A textual or multimedia depiction of the Entity No duplicates allowed in the entity traits list.
ADMINISTRATIVE_GENDER	CE	A value representing the gender (sex) of a Living subject No duplicates allowed in the entity traits list.
BIRTH_TIME	TS	The date and time of a living subject's birth or hatching.

		No duplicates allowed in the entity traits list.
DECEASED_IND	BL	An indication that the subject is dead. No duplicates allowed in the entity traits list.
DECEASED_TIME	TS	The date and time that a living subject's death occurred. No duplicates allowed in the entity traits list.
MULTIPLE_BIRTH_IND	BL	An indication as to whether the living subject is part of a multiple birth. No duplicates allowed in the entity traits list.
MULTIPLE_BIRTH_ORDER_NUMBER	INT	The order in which this living subject was born if part of a multiple birth. No duplicates allowed in the entity traits list.
ORGAN_DONOR_IND	BL	An indication that the living subject is a candidate to serve as an organ donor. No duplicates allowed in the entity traits list.
MARITAL_STATUS	CE	A value representing the domestic partnership status of a person. No duplicates allowed in the entity traits list.
EDUCATION_LEVEL	CE	The highest level of education a person achieved (e.g. elementary school, high school or secondary school degree complete, college or baccalaureate degree complete). No duplicates allowed in the entity traits list.
RACE	CE	A value representing the race of a person. No duplicates allowed in the entity traits list.
DISABILITY	CE	A value identifying a person's disability. No duplicates allowed in the entity traits list.
LIVING_ARRANGEMENT	CE	A value specifying the housing situation of a person. No duplicates allowed in the entity traits list.
RELIGIOUS_AFFILIATION	CE	The primary religious preference of a person (e.g. Hinduism, Islam, Roman Catholic Church). No duplicates allowed in the entity traits list.
ETHNIC_GROUP	CE	The ethnic group of the person. No duplicates allowed in the entity traits list.
PHONE	TEL	A telecommunication address for the Entity; only value field should be used.

MOBILE	TEL	No duplicates allowed in the entity traits list.
FAX	TEL	
EMAIL	TEL	
NAME_FAMILY	ST	<p>A non-unique textual identifier or moniker for the Entity; only text field should be used.</p> <p>No duplicates allowed in the entity traits list.</p>
NAME_FIRST_GIVEN	ST	
NAME_SECOND_GIVEN	ST	
NAME_PREFIX	ST	
NAME_SUFFIX	ST	
HOME_ADDRESS	AD	
PRIMARY_HOME_ADDRESS	AD	
VACATION_HOME_ADDRESS	AD	<p>The postal and/or residential address of a Person; only text field should be used.</p> <p>No duplicates allowed in the entity traits list.</p>
WORK_PLACE_ADDRESS	AD	
DIRECT_ADDRESS	AD	
PUBLIC_ADDRESS	AD	
TEMPORARY_ADDRESS	AD	<p>An identifier that uniquely identifies a thing or object. Examples are object identifier for HL7 RIM objects, medical record number, order id, service catalog item id, Vehicle Identification Number (VIN), etc. Instance identifiers are defined based on ISO object identifiers.</p> <p>There can be duplicates in the entity traits list with this identifier.</p>
UID	II	

4.2.6. Fault contracts

Fault contracts define which errors are raised by the service, and how the service handles and propagates errors to its clients. The following table explains what exception MPI can throw

Class	Description
CustomException	Not used in MPI.
DataTypeUnknown	Exception thrown when a trait has a data type.
DomainIdentifierDoesNotExist	Exception thrown when a domain identifier is not known to the system
EntityIdentifierDoesNotExist	Exception thrown when an entity identifier is unknown to the system.
EntityIdentifierMustBeUniqueForEntityTypeOnDomain	Not used in MPI. EIS compliant.
EntityIdentifierNotAssignedToEntityTypeOnDomain	Exception thrown when an entity identifier is not assigned to an entity type on domain.
EntityIsActive	Exception thrown when trying to activate an entity that is already active.
EntityIsInactive	Exception thrown when trying to inactivate an entity that is already inactive.
EntityTypeClassifierIdentifierDoesNotExist	Exception thrown when an Entity type classifier is unknown to the system.
EntityTypeIdentifierDoesNotExist	Exception thrown when an entity type identifier is unknown to the system.
EntityTypeNotAssignedToDomain	Exception thrown when an entity type is not assigned to a domain
FunctionForMatchAlreadyExists	Exception thrown when trying to add a function for matching an automatic link that already exists.
FunctionForMatchDoesNotExists	Exception thrown when trying to add a new function for match that does not exists in the database.

FunctionForMatchIsInUseByLinkAlgorithm	Exception thrown when trying to delete a function that is in use by a link algorithm
FunctionForMatchUnknown	Exception thrown when trying to add a trait to a link algorithm with a function that is not registered.
IdentifierMustBeUnique	Exception thrown when trying to add an identifier that is already in use.
InsufficientTraits	Exception thrown when creating a new entity and not supplying the mandatory traits for that entity type on domain.
InvalidDataTypeForTrait	Exception thrown when a trait has a data type that does not correspond with the data type with which the trait has been defined.
InvalidValueForTrait	Exception thrown when a trait has a value that is not valid
MatchingAlgorithmNotSupported	Exception thrown when a matching algorithm is not supported.
MismatchBetweenNumberOfDomainAndMandatoryIndicators	Exception thrown when assigning an entity type to more domains not specifying the same number of mandatory indicators as the number of domains
NullMethodParameterOrNoValueStored	Exception thrown when a trait identifier is not known to the system.
TraitListDuplicates	Exception thrown when there are duplicates in the trait list of an entity.
TraitNotAssignedToDomain	Exception thrown when a trait is not assigned to a domain
TraitNotAssignedToEntityType	Exception thrown when a trait is not assigned to an entity type
TraitNotAssignedToEntityTypeOnDomain	Exception thrown when a trait is not assigned to an entity type on domain
UnexpectedError	Exception thrown when an unexpected error occurs.

4.2.7. Database layer

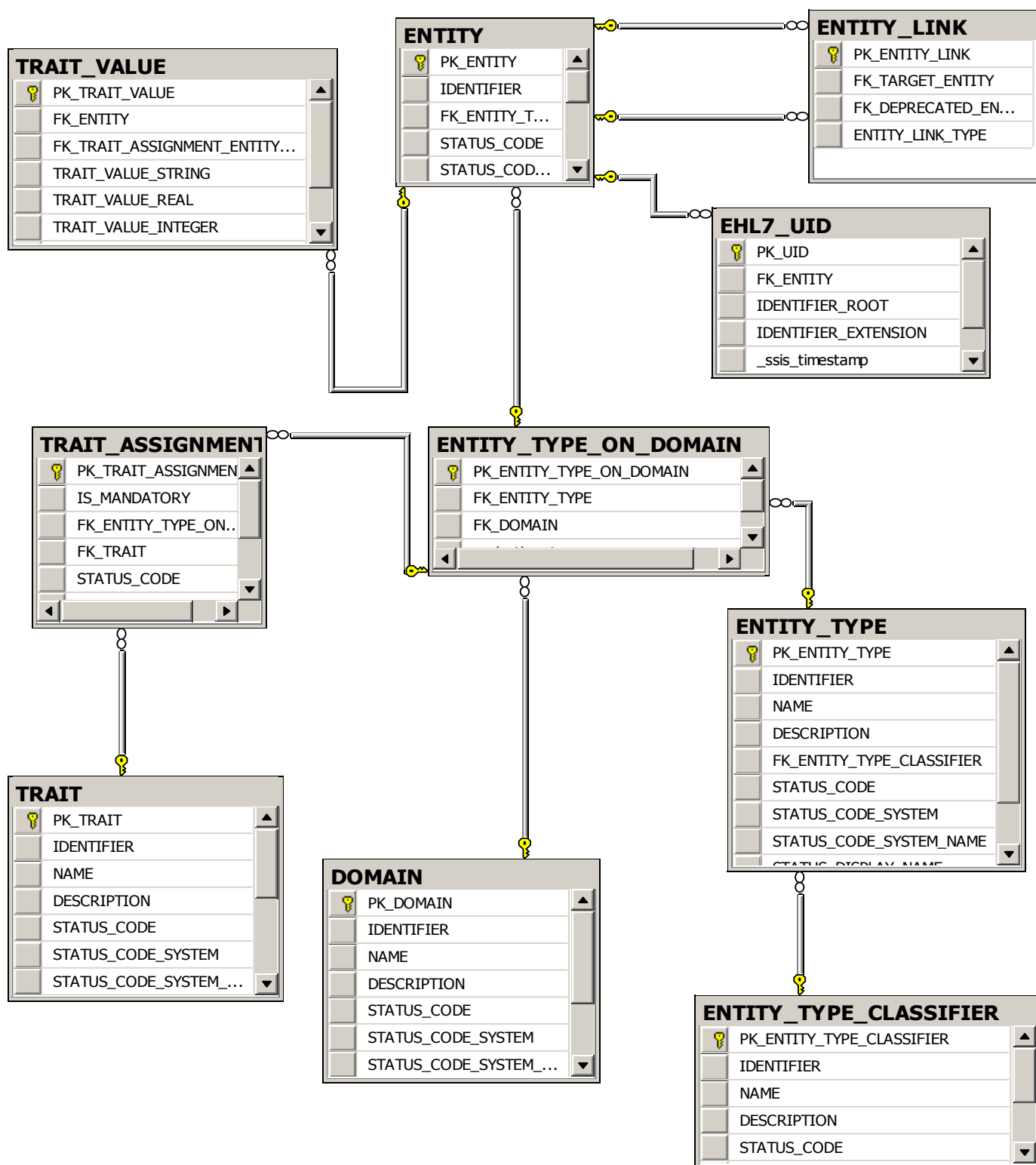


Figure 19 – MPI main database layer

The database tables are somewhat similar with the main MPI classes described earlier in the data contract section.

ENTITY table holds the patient identifier and status code of the entity.

EHL7_UID table is linked with a entity and it stores and HL7 II (Instance Identifier)

ENTITY_LINK table holds information for linked patients (similar patients).

ENTITY_TYPE table is similar to the EntityType class, holding information of what type of entity we are dealing with. For patient, the type is "HL7-RIM-V3-Person".

ENTITY_TYPE_CLASSIFIER table stores one row: "This classifier identifiers entity types that are specializations of the HL7 RIM V3 'Entity' class".

ENTITY_TYPE_ON_DOMAIN table stores the link between entity types and domains.

DOMAIN table hold information about current domains. The active one is "ReMINE".

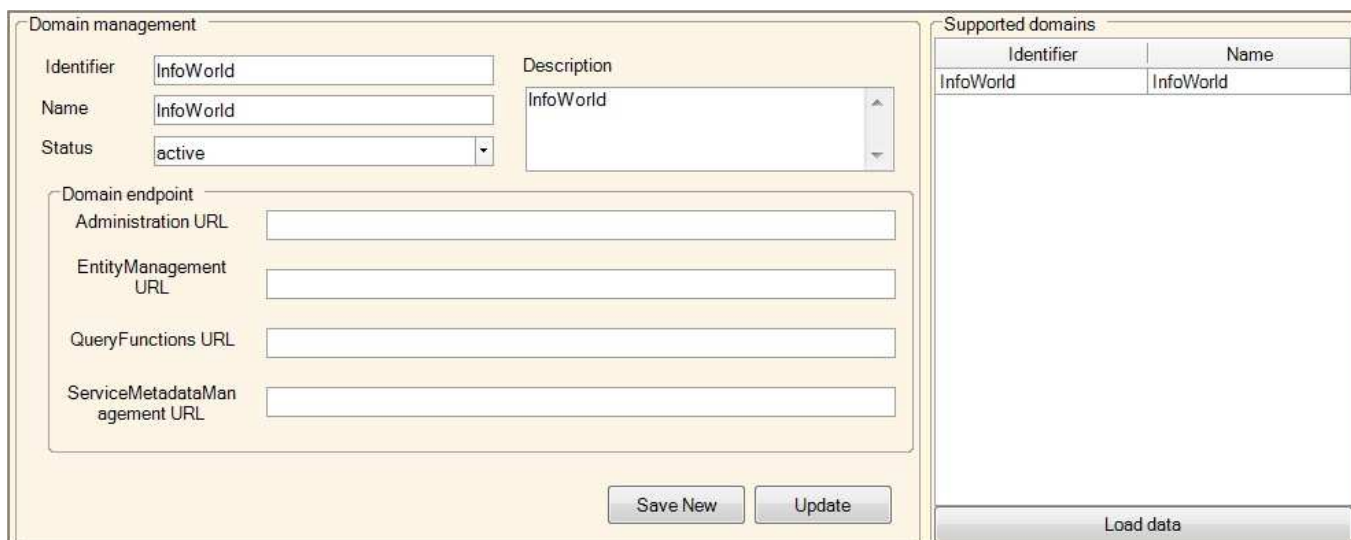
TRAIT table contains a list of all traits (similar to an above table stating patient traits).

TRAIT_ASSIGNMENT table links a TRAIT with an entity type and domain.

TRAIT_VALUE table contains actual values of traits assigned to an entity (patient)

4.2.8. Service metadata management

Creation of domains, traits and trait assignment is done through a MPI service metadata management console. This is a snap-in for Microsoft Management Console that connects to IServiceMetadataManagement interface of MPI service. This console can be used by qualified personnel to alter MPI metadata as described below.



The screenshot displays the MPI service metadata management console. It is divided into two main sections: "Domain management" on the left and "Supported domains" on the right.

Domain management section:

- Identifier:** InfoWorld
- Name:** InfoWorld
- Status:** active (dropdown menu)
- Description:** InfoWorld
- Domain endpoint section:**
 - Administration URL: [empty text box]
 - EntityManagement URL: [empty text box]
 - QueryFunctions URL: [empty text box]
 - ServiceMetadataManagement URL: [empty text box]
- Buttons:** "Save New" and "Update"

Supported domains section:

Identifier	Name
InfoWorld	InfoWorld

At the bottom of the "Supported domains" section is a "Load data" button.

Figure 20 – MPI CreateDomain / UpdateDomainDefinition

Entity type classifier management

Identifier:

Name:

Status:

Description:

Save New

Update

Supported entity type classifiers

Identifier	Name
HL7-RIM-V3-Entity	HL7 RIM V3 Entity sub types

Load data

Figure 21 – MPI Entity type classifier management

Entity type management

Identifier:

Name:

Status:

Classifier:

Description:

Save New

Update

Supported entity types

Identifier	Name
HL7-RIM-V3-Person	HL7-RIM-V3-Person
HL7-RIM-V3-NonPersonLiving...	HL7-RIM-V3-NonPersonLivingSubject
HL7-RIM-V3-Place	HL7-RIM-V3-Place
HL7-RIM-V3-Organization	HL7-RIM-V3-Organization
HL7-RIM-V3-Material	HL7-RIM-V3-Material
HL7-RIM-V3-ManufacturedMa...	HL7-RIM-V3-ManufacturedMaterial
HL7-RIM-V3-Container	HL7-RIM-V3-Container
HL7-RIM-V3-Device	HL7-RIM-V3-Device

Load data

Figure 22 – MPI Entity type management – insert/update

D.3.3 First Revision Data & Process model system framework

Trait management

Identifier: ADMINISTRATIVE_GENDER
Name: Administrative_Gender
Status: active
DataType: CE
Description:

Save New
Update

Supported traits

Identifier	Name	DataType
CODE	Code	CS
QUANTITY	Quantity	PQ
DESCRIPTION	Description	ED
ADMINISTRATIVE_GENDER	Administrative_Gender	CE
BIRTH_TIME	BIRTH_TIME	TS
DECEASED_IND	Deceased_Ind	BL
DECEASED_TIME	Deceased_Time	TS
MULTIPLE_BIRTH_IND	Multiple_Birth_Ind	BL
MULTIPLE_BIRTH_ORDER_NUMBER	MULTIPLE_BIRTH_ORDER_NUMBER	INT
ORGAN_DONOR_IND	ORGAN_DONOR_IND	BL
NAME_FAMILY	NAME_FAMILY	ST
NAME_FIRST_GIVEN	NAME_FIRST_GIVEN	ST
NAME_SECOND_GIVEN	NAME_SECOND_GIVEN	ST
NAME_PREFIX	NAME_PREFIX	ST
NAME_SUFFIX	NAME_SUFFIX	ST
MARITAL_STATUS	MARITAL_STATUS	CE
EDUCATION_LEVEL	EDUCATION_LEVEL	CE
RACE	RACE	CE
DISABILITY	DISABILITY	CE
LIVING_ARRANGEMENT	LIVING_ARRANGEMENT	CE
RELIGIOUS_AFFILIATION	RELIGIOUS_AFFILIATION	CE
ETHNIC_GROUP	ETHNIC_GROUP	CE
MOBILE_IND	MOBILE_IND	BL
DIRECTIONS_TEXT	DIRECTIONS_TEXT	ED

Load data

Figure 23 – MPI Trait management– insert/update

Domain: InfoWorld

Assigned entity types

IDENTIFIER	NAME
*	
HL7-RIM-V3-Container	HL7-RIM-V3-Container
HL7-RIM-V3-Device	HL7-RIM-V3-Device
HL7-RIM-V3-ManufacturedMaterial	HL7-RIM-V3-ManufacturedMaterial
HL7-RIM-V3-Material	HL7-RIM-V3-Material
HL7-RIM-V3-NonPersonLivingSubject	HL7-RIM-V3-NonPersonLivingSubject
HL7-RIM-V3-Organization	HL7-RIM-V3-Organization
HL7-RIM-V3-Person	HL7-RIM-V3-Person
HL7-RIM-V3-Place	HL7-RIM-V3-Place

Save assignments

Figure 24 – MPI Entity Type Assignment management

Entity Type:

Assigned traits		
Identifier	DataType	IsMandatory
*		
ADMINISTRATIVE_GENDER	CE	<input type="checkbox"/>
ADTV	R_Additive	<input type="checkbox"/>
AFFL	R_Affiliate	<input type="checkbox"/>
AGNT	R_Agent	<input type="checkbox"/>
ASSIGNED	R_AssignedEntity	<input type="checkbox"/>
BIRTH_TIME	TS	<input type="checkbox"/>
BIRTHPL	R_BirthPlace	<input type="checkbox"/>
CAREGIVER	R_Caregiver	<input type="checkbox"/>
CATEGORIE_ASIGURAT_CODE	ST	<input type="checkbox"/>
CATEGORIE_ASIGURAT_CODESYSTEM	ST	<input type="checkbox"/>
CATEGORIE_ASIGURAT_CODESYSTEMNAME	ST	<input type="checkbox"/>
CATEGORIE_ASIGURAT_DISPLAYNAME	ST	<input type="checkbox"/>
CETATENIE_CODE	ST	<input type="checkbox"/>
CETATENIE_CODESYSTEM	ST	<input type="checkbox"/>
CIT	R_Citizen	<input type="checkbox"/>
COMPAR	R_CommissioningParty	<input type="checkbox"/>
CON	R_Contact	<input type="checkbox"/>
CONTACT1	TEL	<input type="checkbox"/>
CONTACT2	TEL	<input type="checkbox"/>

[Save assignments](#)

Figure 25 – MPI Trait Assignment management

5. Process mapper

5.1. General Description

The process mapping is the graphical representation of the processes inside an organisation. The process map shows all steps and the activities together with inputs and outputs. The scope of the process mapping is to provide a unifying vision of business processes, so that the organisation and the individuals have a common understanding. In ReMINE project the Process Mapper is used for the modeling of the processes of a healthcare organisation. For this application is used a client/server tool, the Intalio Designer¹. This module will be used by BPM experts and the responsible healthcare organisation managers (risk managers) once per ReMINE system deployment.

The scope of the ReMINE project is to predict, detect and manage Risks Against Patient Safety (RAPS). For the ReMINE four pilots, different scenarios were identified for each one, and these scenarios were mapped with the use of BPMN (Business Process Modeling Notation). For the ReMINE project these processes are the basis for the identification and the prediction of risks. The “Process Mapper” serves two main functions the mapping of hospital’s processes and the association of those with data and Data Events. First the processes are modeled in BPMN format, then are transformed in BPEL format for being executable and be able to trigger the appropriate web services and finally are associated to data and Data Events and are enriched with KPIs (Key Performance Indicators). The use of this methodology entails a dynamic system for inserting data, receiving messages, for associating with Data Events and according to these having as output risk identification/prediction and alert triggering for risk management (corrective actions). This

¹ <http://www.intalioworks.com/products/bpm/opensource-edition/designer/>

module is dependent from the ReMINE sub-module “RAPS Management System” where the execution of the processes (BPEL transformation) is implemented.

The scenarios that are used were defined by the hospital personnel and the risks for these were also identified. The corresponding processes are representing the situation in each hospital's specific Unit (A&E (Accident and Emergency) Unit for Niguarda, Obstetrics and Gynecology Unit for Sacco and Acute Care for Elderly for Kauhajoki healthcare centre) and describe the actions, the interactions between the users, or between the users and the system or the decisions that must be taken inside each hospital etc. For the fourth pilot Rotherham NHS Foundation Trust (this scenario deals with the infection control in the medical admission ward) the scenario is being updated after hospital's personnel request and for this reason the corresponding processes will be implemented in the next months.

In the next paragraphs are described in detail the translation of the BPMN diagrams in fully executable BPEL processes and the association of the processes with Data. The descriptions are detailed for each pilot.

5.2. Process Mapper for Sacco

The scenario focuses on the FHR monitoring activity during labour and delivery assistance: it considers the risk for patient safety due to a delayed execution of clinical protocol, because this task provides medical staff with important information that enable a correct assessment and the selection of the best clinical pathway.

The scenario description and the steps that were used in process modeling are described in the D7.2- Pilots Test Environment Definition. The primary users of this scenario are the nurses and obstetricians that are in charge of filling the forms.

The steps modeled in the Sacco process are the following:

Diagnosis of active and low risk labour

- If new woman arrives at Obstetric A&E
 - Midwife fills in the A&E report (“Isolabella”) with anagraphic data.
 - Midwife creates the barcode bracelet and associates it to the woman.
 - Midwife applies the barcode bracelet on the woman.
 - Obstetrician on duty is called.
 - Obstetrician fills in the A&E report and the REMINE forms (checklists) for admission
 - Midwife prepares the woman for the first FHR monitoring.
 - First FHR monitoring is carried out (see “FHRmonitoringexecution” steps).
 - Obstetrician makes the diagnosis and decides about the hospitalization.

FHR monitoring execution

- Midwife approaches the woman with the EFM device

- Midwife prepares the woman for the monitoring
- Midwife activates the registration of the FHR monitoring through the scanning of:
 - The woman's bracelet.
 - Her own barcode bracelet.
 - The EFM device barcode label
- When the FHR monitoring is finished, the midwife stops the registration of the FHR monitoring through the scanning of:
 - The woman's bracelet.
 - Her own barcode bracelet.
 - The EFM device barcode label

Obstetrician assessment

- Obstetrician registers the results of the assessment into the REMINE forms.

"Exit"

- If the baby is born or if woman is discharged or transferred to another ward
 - Obstetrician has to fill in the REMINE "Exit" form
 - REMINE registers that the specific woman left the clinical pathway.

For the Sacco process two BPM Intalio Projects have been created, one to manage the business part, Sacco Project, and the other one to manage the data, Database Manager Project. It is not necessary to separate them but in this way the diagrams are more clear and easier to understand and even to model the process is less complex.

The database manager will be explained in future documents because is still not completed and all the tables and the data inside the tables are just an approximation to the final model.

About the Sacco Project, it contains four BPDs, three of them are templates that are using by the main one, the main one is called Sacco.bpm. In the next screenshot the four diagrams are shown.

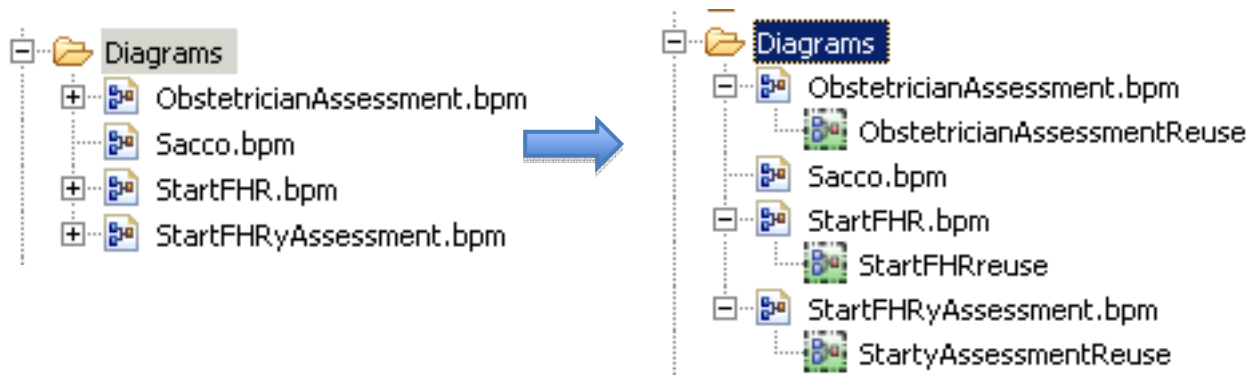


Figure 26 – Process Explorer for Sacco Diagrams

Sacco BPD

Sacco BPD has five pools inside:

- DEMS: is the pool that starts the process sending a message with the patient for the admission and the rest of the tasks.
- User: in this case the users are not divided and the obstetrician and the nurse share the same pool, therefore for this diagram the user is the person that interacts with the computer.
- REMINE: this pool represents the system.
- Server: it is useful to talk with the Intalio server, for the moment is useless.
- Database: is the pool that sends the orders to the database.

The diagram is as follows:

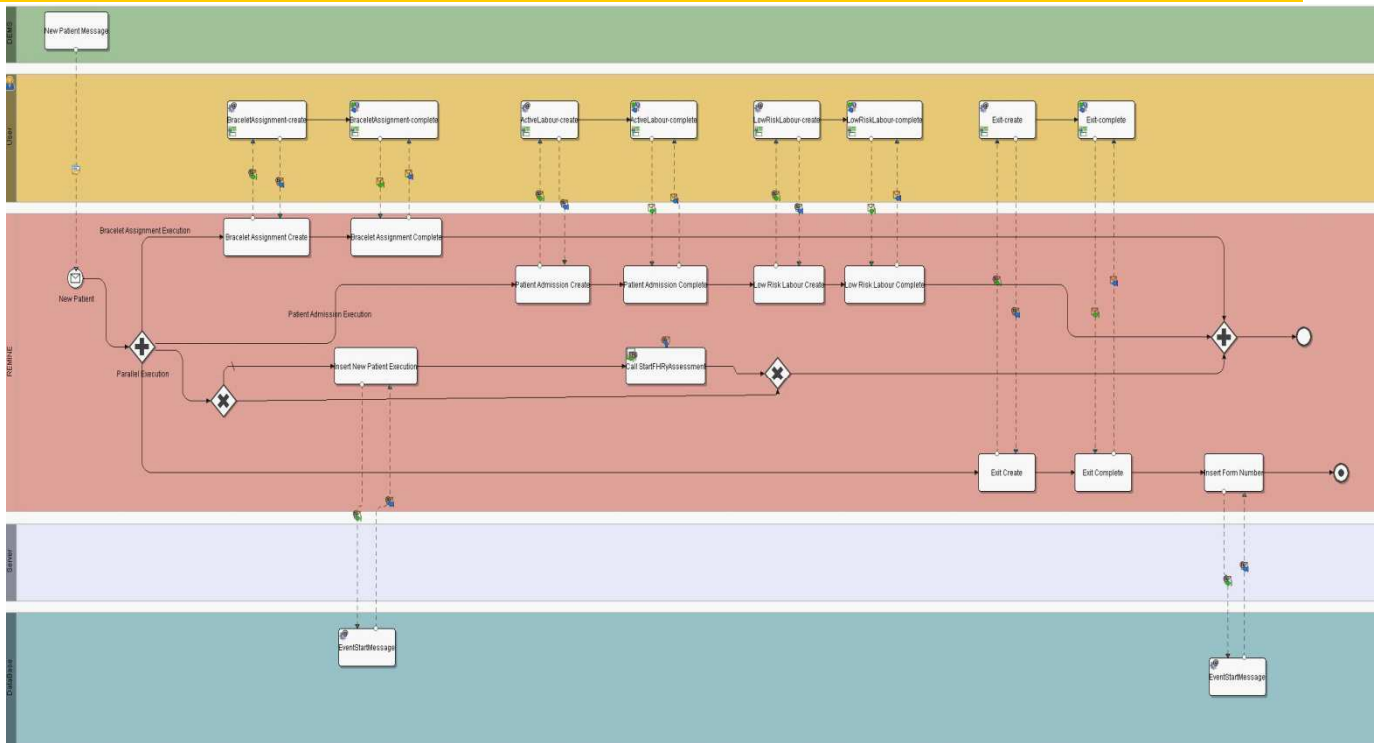


Figure 27 – Sacco BPD

Because the diagram is very big we have decided to split the diagram in some chunks in order to describe it step by step

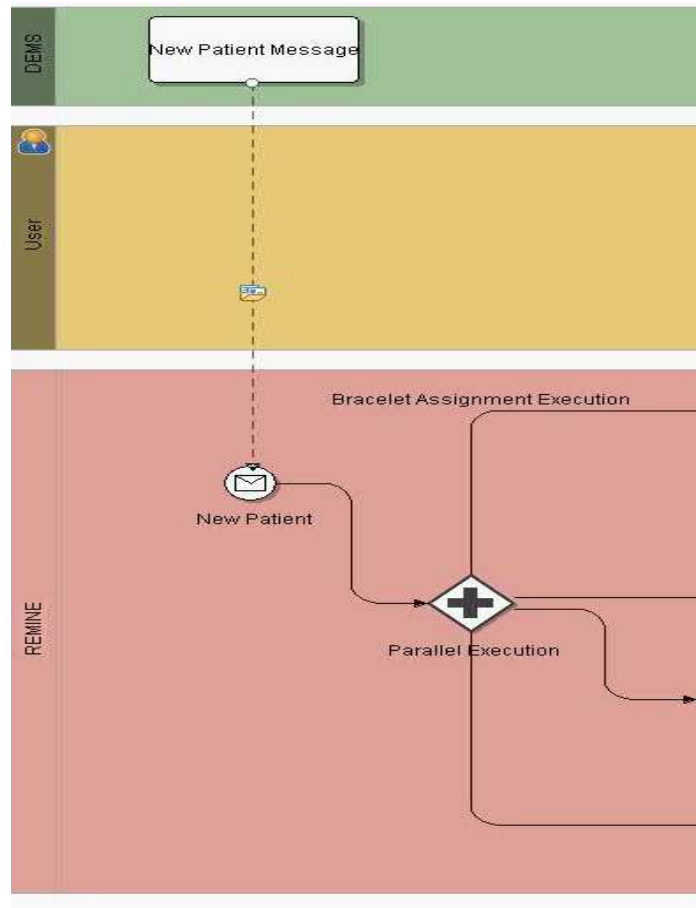


Figure 28 – Start and Parallel Gateway

DEMS sends a message with the new patient and the process starts. Then there is a parallel gateway that provokes the execution of four branches at the same time:

1. Bracelet Assignment: it creates the form for the bracelet assignment and receives the data filled in by the nurse.

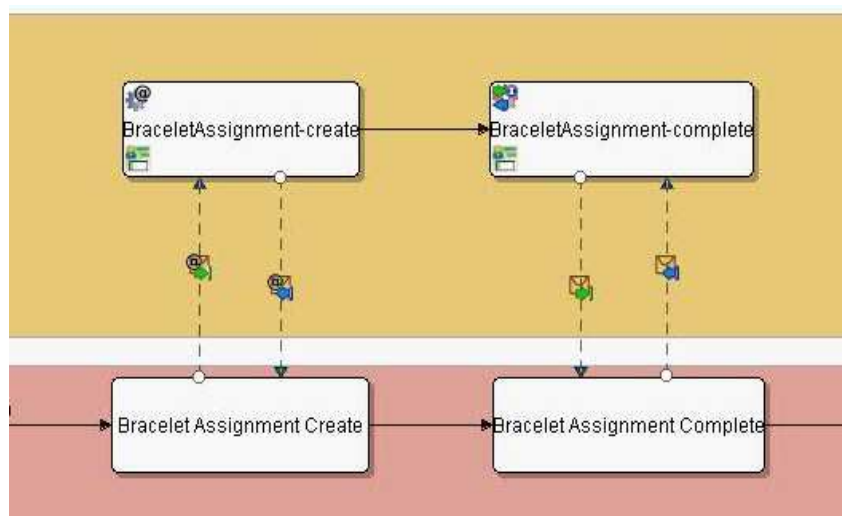
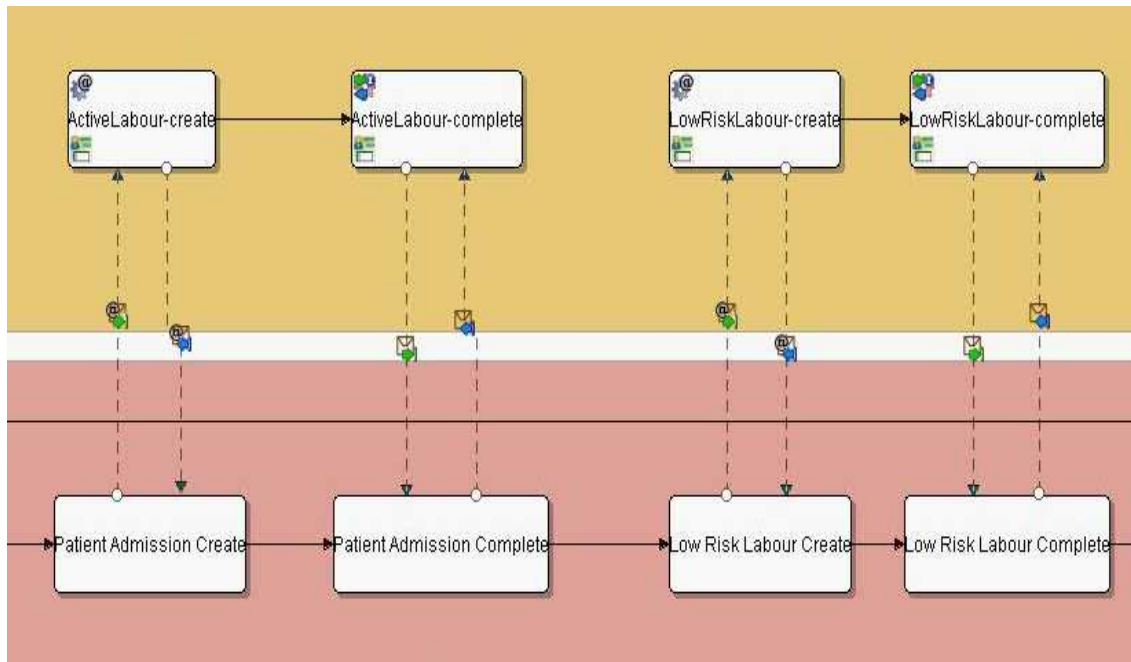


Figure 29 – Bracelet Assignment

2. Patient Admission: two chained forms are created and completed, the first one is about the “Active Labour” and the second is about the “Low Risk Labour”.

**Figure 30 – Active and Low Risk Labour**

3. Obstetrician Assessment and FHR Monitoring call: this third branch inserts the patient in the internal database in order to control the execution of the process, then it calls the “StartFHRyAssessment” template.

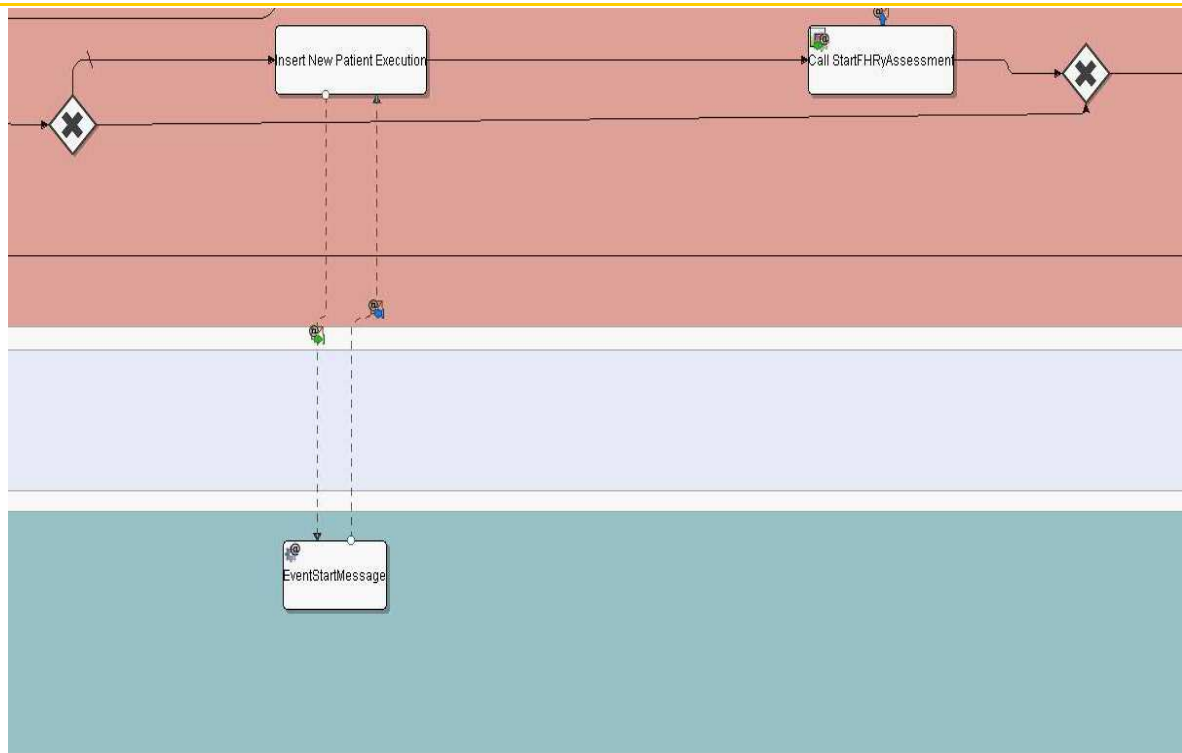


Figure 31 – Obstetrician Assessment and FHR Monitoring Call

4. Exit: it shows the “Exit” to be filled in by the obstetrician and inserts a number in the database to indicate that the execution for that patient is finished.

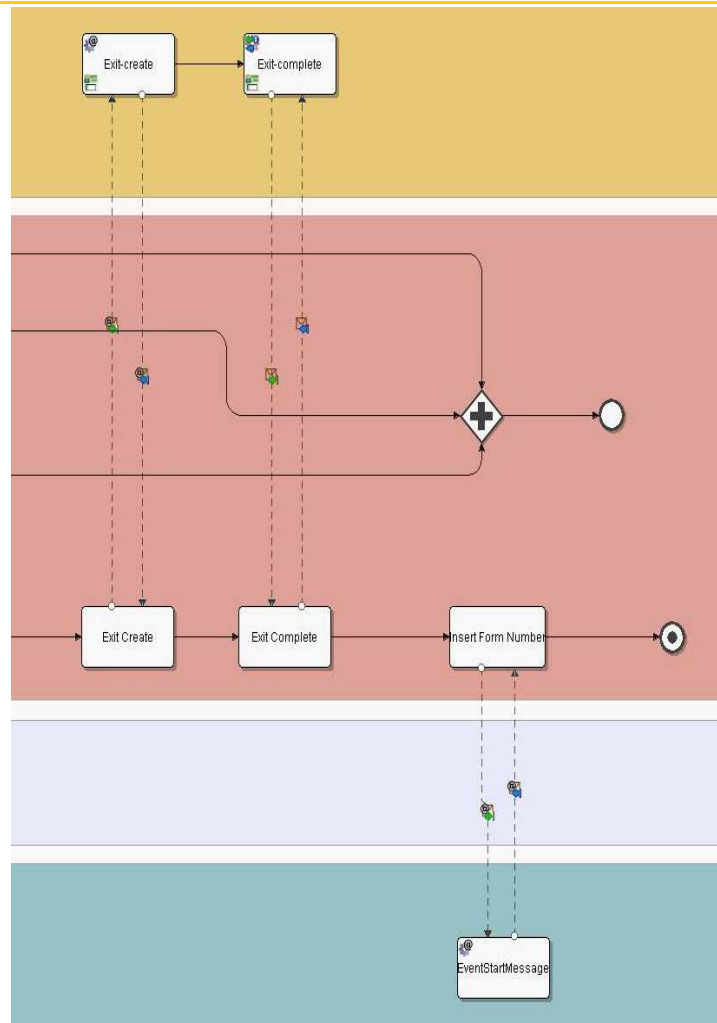


Figure 32 – Exit

At the end all the branches are collected through a gateway and the process ends.

StartFHRyAssessment BPD

StartFHRyAssessment BPD has three pools inside:

- Invoker: is the actor that calls this process.
- System: this pool represents the system.
- Database: is the pool that sends the orders to the database.

The diagram is as follows:

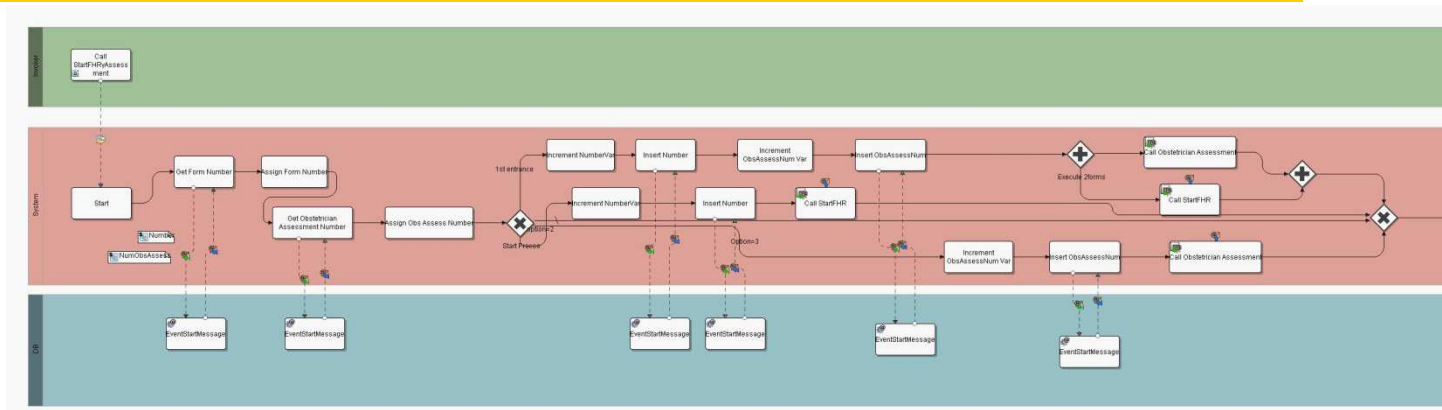


Figure 33 – StartFHRyAssessment BPD

As in the previous BPD we are going to divide the picture in chunks to explain them step by step.

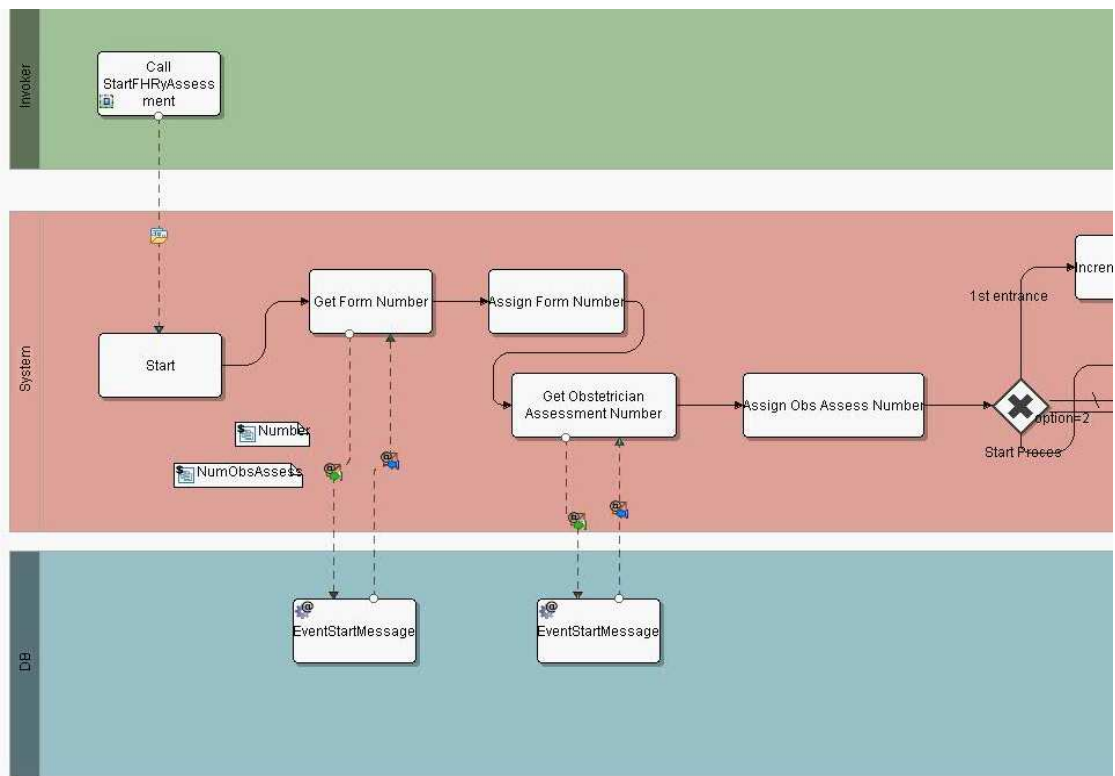


Figure 34 – Start

In this first figure it is exposed that the process is called and starts from the “Start” task and after that task there is one call to the database to get the number of forms of the “Start FHR monitoring” that has been filled in because this number is stored in the database, the next task is “Assign Form number” that is in charge of assign the number of forms to the variable called “Number”. The next two tasks do the same thing but with the number of “Obstetrician Assessment” forms (“NumObsAssess” variable).

Finally there is an Exclusive Data-Based Gateway, that will execute one of these three branches:

1. First Entrance: this branch is executed when it is the first time that the process is called for a patient, this time the process is called by the Sacco BPD. The task “Increment NumberVar” increments the variable “Number” by one and the second tasks insert the value of the variable in the database. The two following tasks do exactly the same but with the variable “NumObsAssess”.

The next element is a parallel gateway that executes the calls to the “Obstetrician Assessment” in the first branch and to the “StartFHR” in the second one.

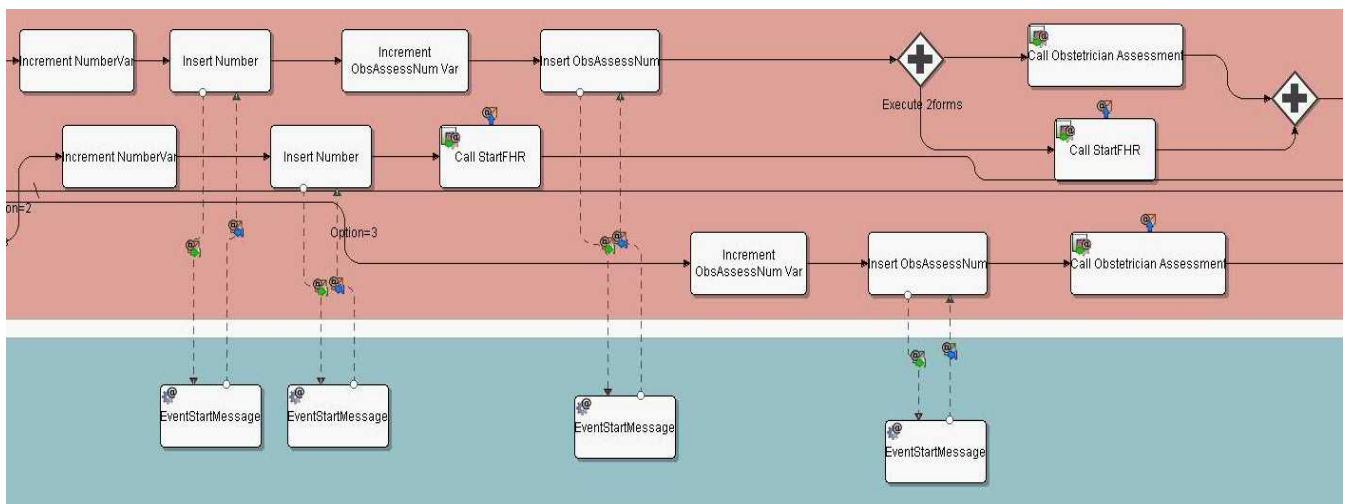


Figure 35 – First Entrance

2. Called by “Start FHR”: this is the second branch and it is executed when the call comes from the “Start FHR” BPD. It increments the variable “Number” in the first task and stores it in the second task, after that it does a call to the “Start FHR” process. (This branch is visible in the previous figure)
3. Called by “Obstetrician Assessment”: this is the third branch and it is executed when the call comes from the “Obstetrician Assessment” BPD. It increments the variable “ObsAssessNum” in the first task and stores it in the second task, after that it does a call to the “Obstetrician Assessment” process. (This branch is visible in the previous figure)

StartFHR

StartFHR BPD has three pools inside:

- Invoker: is the actor that calls this process.
- System: this pool represents the system.
- User: the person that interacts with the system.

The diagram is as follows:

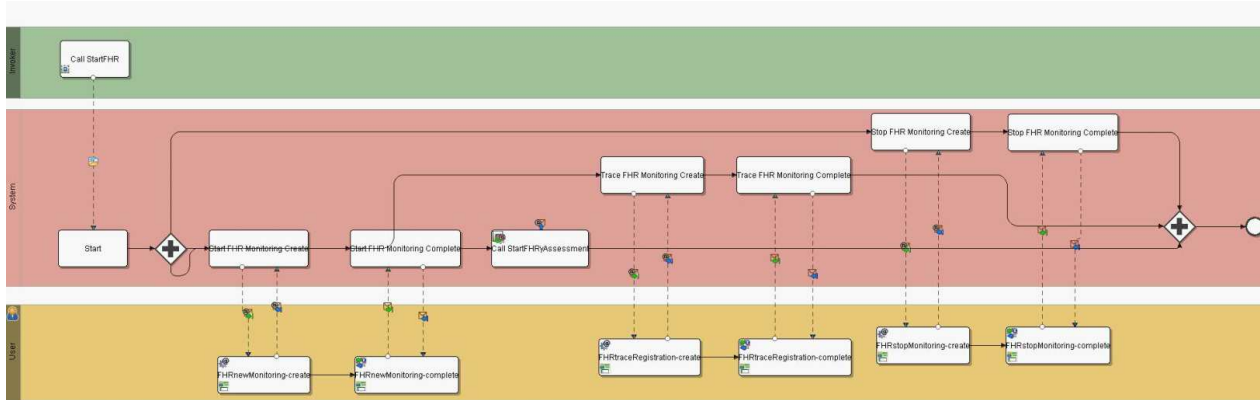


Figure 36 – StartFHR BPD

The process starts with the invocation, just after that a parallel gateway is applied, to execute three branches:

1. Start FHR Monitoring: this part of the process creates and completes the “Start FHR Monitoring” form, and calls the “StartFHRyAssessment” template. This call is going to be useful to start with the “StartFHR” again, it means that the template can not call itself, it needs to call another process to come back again.

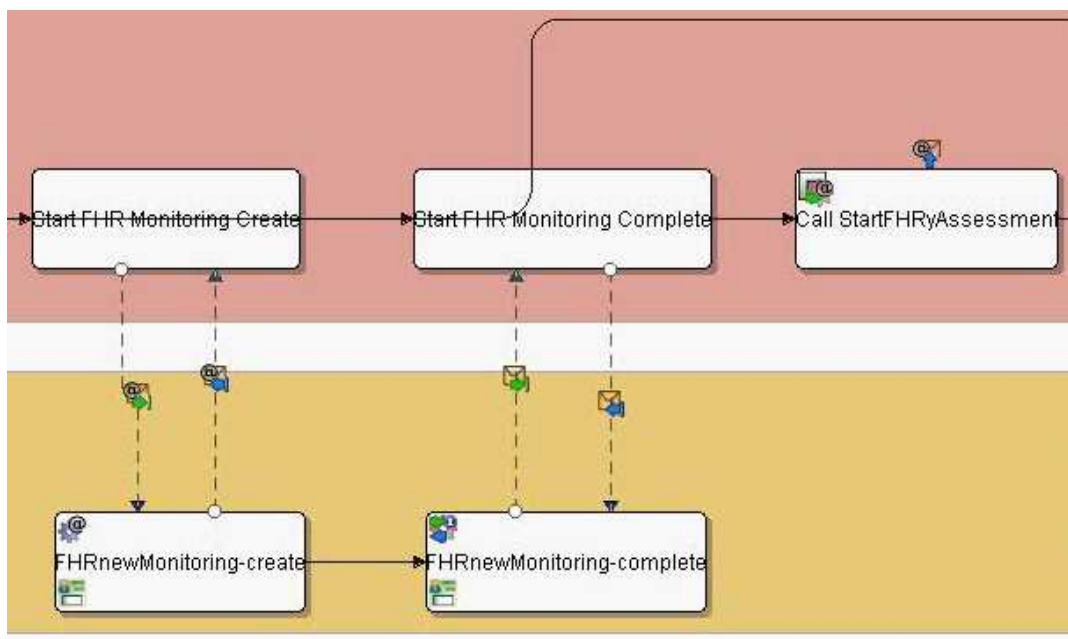


Figure 37 – Start FHR Monitoring

2. Trace FHR Monitoring: this part of the process creates and completes the “Trace FHR Monitoring” form.

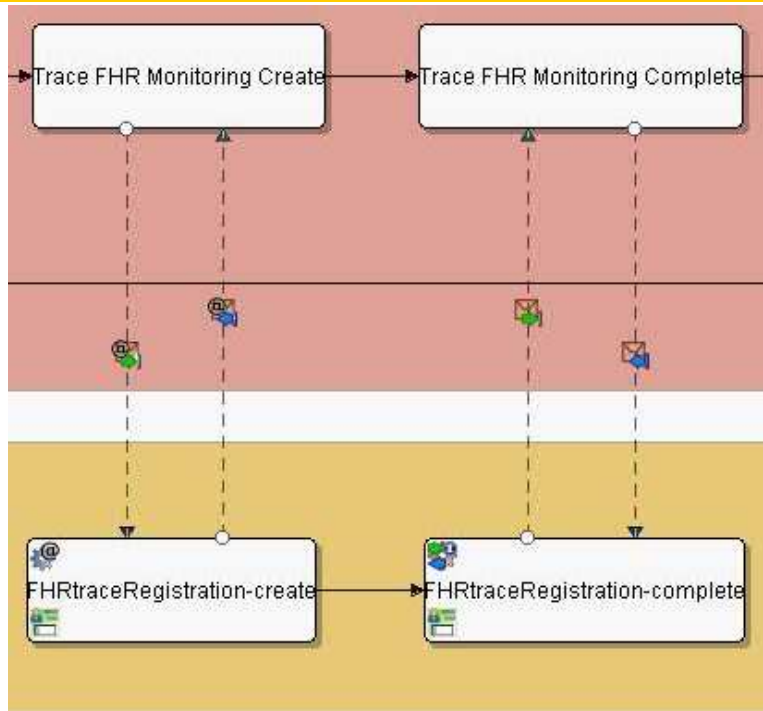


Figure 38 – Trace FHR Monitoring

3. Stop FHR Monitoring: this part of the process creates and completes the “Stop FHR Monitoring” form.

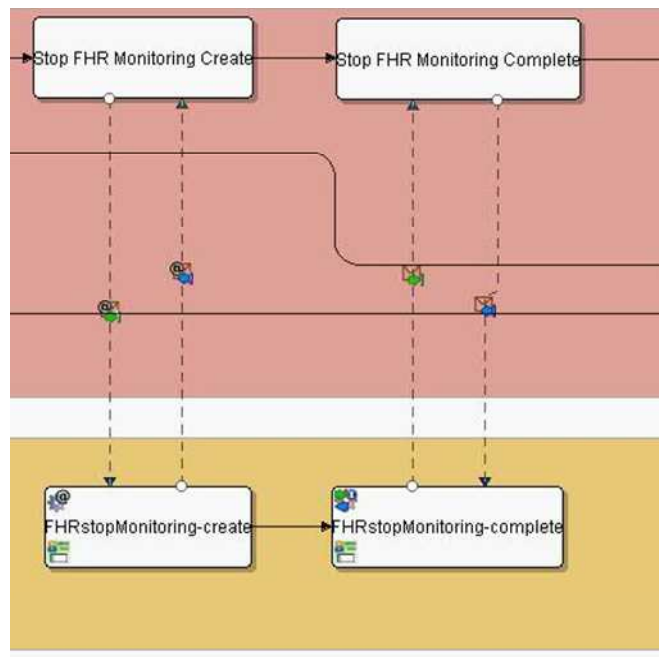


Figure 39 – Stop FHR Monitoring

Obstetrician Assessment

Obstetrician Assessment BPD has three pools inside:

- Invoker: is the actor that calls this process.
- System: this pool represents the system.
- User: the person that interacts with the system.

The diagram is as follows:

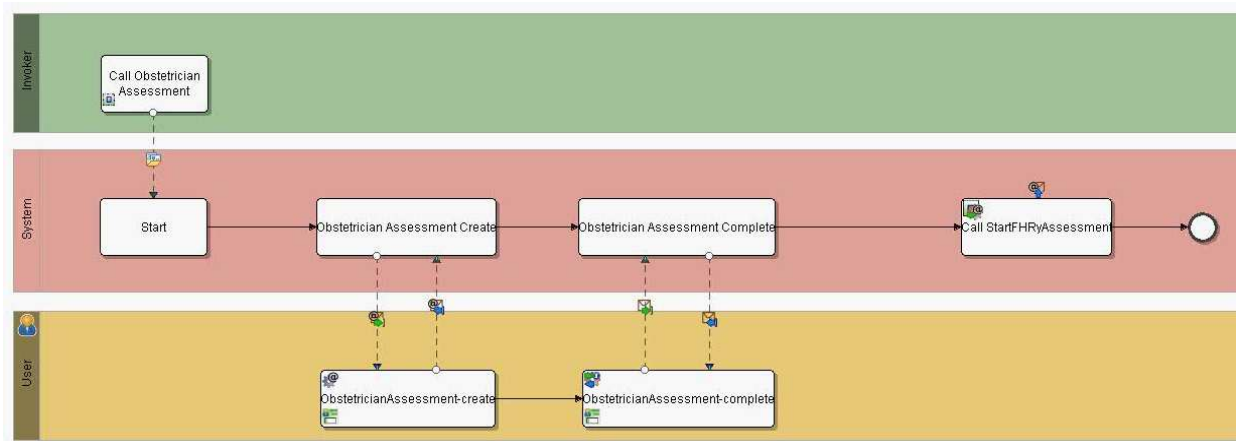


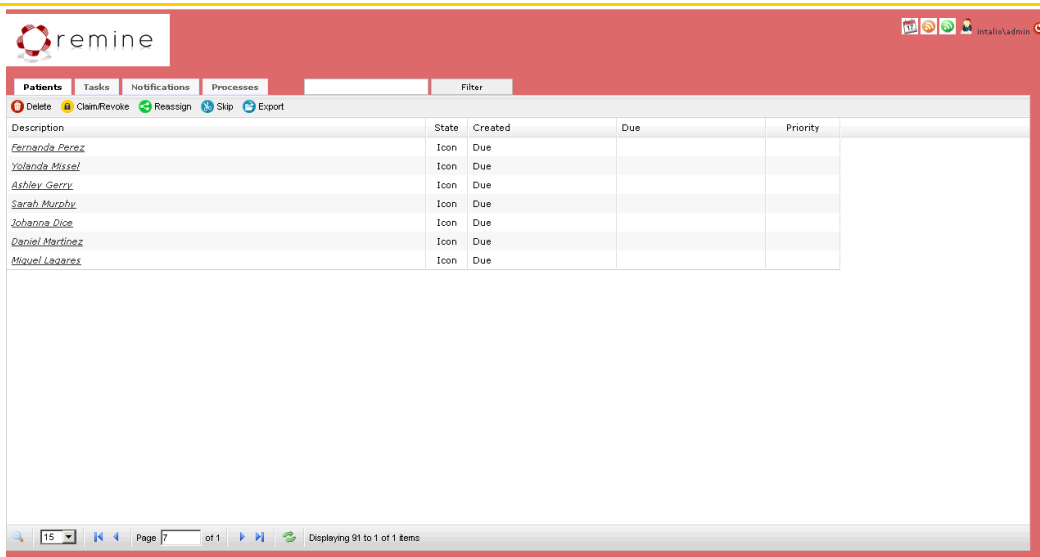
Figure 40 –Obstetrician Assessment BPD

The process creates and completes the "Obstetrician Assessment" form and then it calls the "StartFHRyAssessment" process. It calls that process because the same reason as the last BPD, to be called again later on.

5.2.1 User interface

Patients tab

The 'Patients' tab show all the patients available in the service. By pressing a patient, the 'Task' tab will be opened offering the several available tasks for the chosen patient. Its function is to filter the all the tasks to show just the tasks for the selected patient.




Description	State	Created	Due	Priority
Fernanda Perez	Icon	Due		
Yolanda Misse!	Icon	Due		
Ashley Gerry	Icon	Due		
Sarah Murphy	Icon	Due		
Johanna Dice	Icon	Due		
Daniel Martinez	Icon	Due		
Miguel Lagares	Icon	Due		


Figure 41 – Patients tab

Tasks tab

In the initial interface the end-user by pressing the tab 'Tasks', will be able to choose the required task for a given patient. If the 'Tasks' tab is opened directly (without using 'Patients' tab) all the available tasks for all the patients will be shown. Each time a message from DEMS is received with a new patient, seven tasks will be available for the end-user. The description of these tasks has the associated name of the patient and a proper name to be identified by the end-user. And also in the iterative tasks a number is associated:

1. Start FHR Monitoring
2. Trace FHR Monitoring
3. Stop FHR Monitoring
4. Obstetrician Assessment
5. Admission
6. Bracelet Assignment
7. Exit




intalio@admin

Patients

Tasks

Notifications

Processes

Filter

Delete
 Claim/Revoke
 Reassign
 Skip
 Export

Description	State	Created	Due	Priority	Attachments
Laura Holmes - Stop FHR Monitoring 1	✓	26/02/10 19:08			
Laura Holmes - Trace FHR Monitoring 1	✓	26/02/10 19:08			
Laura Holmes - Start FHR Monitoring 1	✓	26/02/10 19:08			
Laura Holmes - Obstetrician Assessment 1	✓	26/02/10 19:08			
Laura Holmes - Admission	✓	26/02/10 19:08			
Laura Holmes- Exit	✓	26/02/10 19:08			
Laura Holmes - Bracelet Assignment	✓	26/02/10 19:08			
John Gates - Trace FHR Monitoring 1	✓	26/02/10 19:07			
John Gates - Start FHR Monitoring 1	✓	26/02/10 19:07			
John Gates - Stop FHR Monitoring 1	✓	26/02/10 19:07			
John Gates - Obstetrician Assessment 1	✓	26/02/10 19:07			
John Gates - Bracelet Assignment	✓	26/02/10 19:07			
John Gates- Exit	✓	26/02/10 19:07			
John Gates - Admission	✓	26/02/10 19:07			


15

Page 1 of 1

Displaying 1 to 14 of 14 items

Figure 42 – Tasks Tab

1. Start FHR Monitoring





 intalio@admin

Patients Tasks Notifications Processes						Filter
FHR monitoring registration						
<div> <div>New Monitoring</div> <div> Midwife <input type="text"/> Patient <input type="text"/> Device <input type="text"/> </div> </div>						
<input type="button" value="Save"/> <input type="button" value="Claim"/> <input type="button" value="Revoke"/> <input type="button" value="Complete"/>						

Figure 43 – Start FHR Monitoring

Here the names or codes of the midwife, patient and device corresponding to this monitoring will be fulfilled. Once this form is completed in the main interface in the tab 'Tasks' 3 new tasks will be created to allow a new FHR monitoring. The description of the tasks will contain a number for their identification. These tasks are: start, trace and stop FHR monitoring.




 intalio/admin

Patients
Tasks
Notifications
Processes
Filter

Delete
Claim/Revoke
Reassign
Skip
Export


Description	State	Created	Due	Priority	Attachments
Laura Holmes - Trace FHR Monitoring 2	✓	2/26/10 7:36 PM			
Laura Holmes - Stop FHR Monitoring 2	✓	2/26/10 7:36 PM			
Laura Holmes - Start FHR Monitoring 2	✓	2/26/10 7:36 PM			
Laura Holmes - Stop FHR Monitoring 1	✓	2/26/10 7:08 PM			
Laura Holmes - Trace FHR Monitoring 1	✓	2/26/10 7:08 PM			
Laura Holmes - Obstetrician Assessment 1	✓	2/26/10 7:08 PM			
Laura Holmes - Admission	✓	2/26/10 7:08 PM			
Laura Holmes - Exit	✓	2/26/10 7:08 PM			
Laura Holmes - Bracelet Assignment	✓	2/26/10 7:08 PM			
John Gates - Trace FHR Monitoring 1	✓	2/26/10 7:07 PM			
John Gates - Start FHR Monitoring 1	✓	2/26/10 7:07 PM			
John Gates - Stop FHR Monitoring 1	✓	2/26/10 7:07 PM			
John Gates - Obstetrician Assessment 1	✓	2/26/10 7:07 PM			
John Gates - Bracelet Assignment	✓	2/26/10 7:07 PM			
John Gates - Exit	✓	2/26/10 7:07 PM			


15
Page 1 of 2
Displaying 1 to 15 of 15 items

Figure 44 – Patient Tasks. After completed a 'Start FHR Monitoring' task.

Three new task are created with a consecutive number in the description for identifying them

2. Trace FHR Monitoring




 intalio/admin

Patients
Tasks
Notifications
Processes
Filter

FHR Trace Registration

FHR Monitoring

End of FHR monitoring
00:00
01.01.2009

Midwife

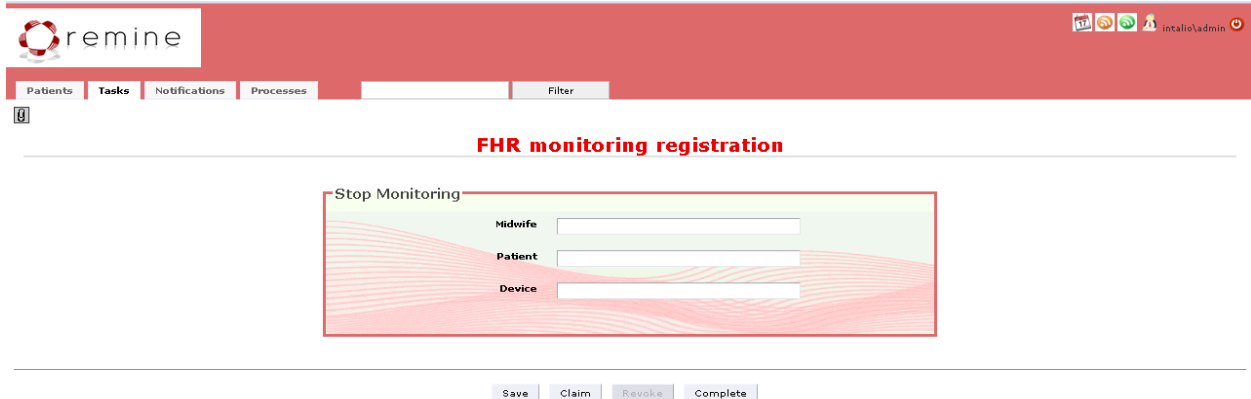
Regular tracing?
☐ Yes
 ☒ No

Save
Claim
Revoke
Complete

Figure 45 – Trace FHR Monitoring

This form is used to score the regular tracing of the FHR monitoring. When the FHR Trace Registration is completed, it disappears from the tasks.

3. Stop FHR Monitoring



The screenshot shows the 'remine' application interface. At the top, there's a navigation bar with 'Patients', 'Tasks', 'Notifications', and 'Processes' tabs. The 'Tasks' tab is active. Below the navigation bar, there's a search bar and a 'Filter' button. The main content area is titled 'FHR monitoring registration'. It contains a 'Stop Monitoring' form with three input fields: 'Midwife', 'Patient', and 'Device'. Below the form, there are four buttons: 'Save', 'Claim', 'Revoke', and 'Complete'.

Figure 46 – Stop FHR Monitoring

This form stops the monitoring associated with the midwife, patient and device introduced in this form. Once is completed, it is removed from the tasks.

4. Obstetrician Assessment

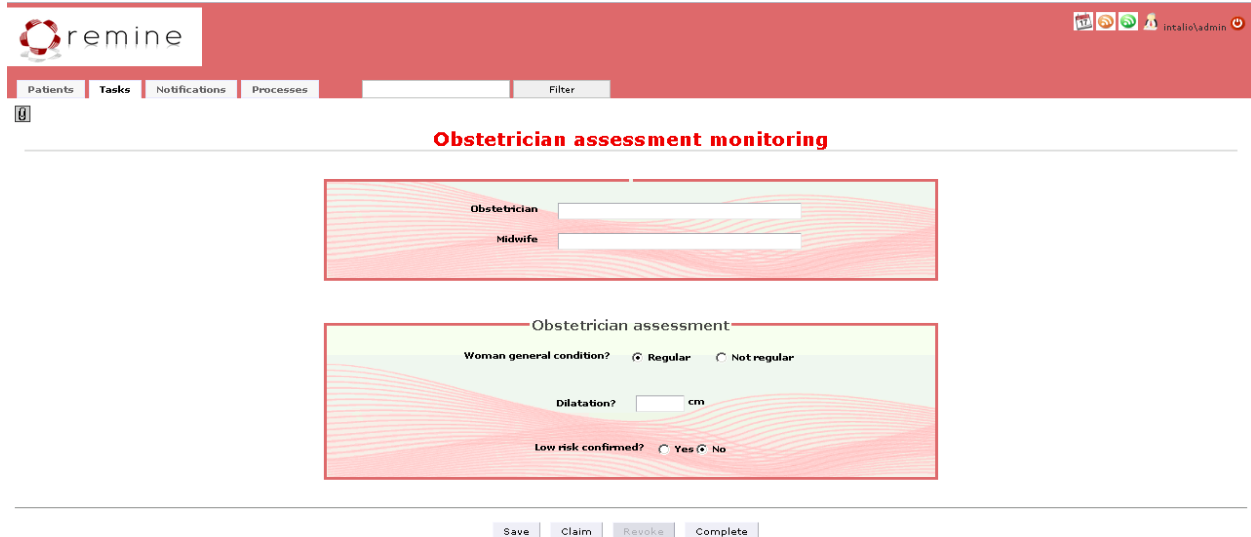
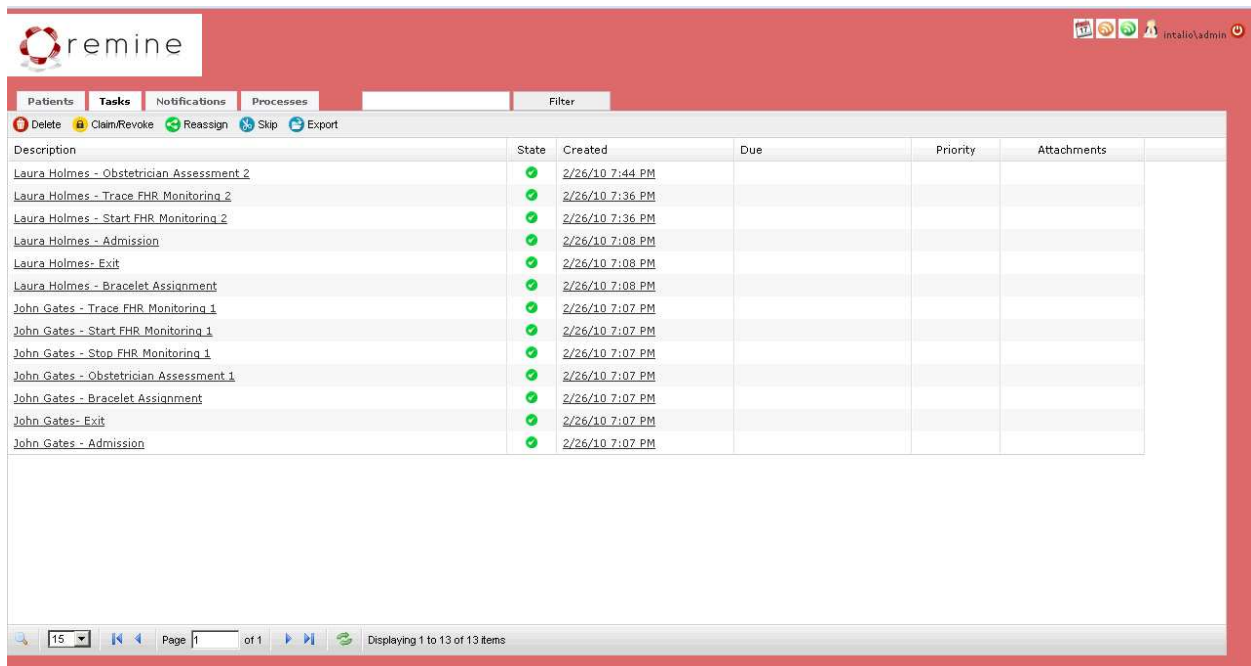


Figure 47 – Obstetrician Assessment

Once this form is completed by the obstetrician, it will be automatically removed from the tasks. Also a new task to perform another obstetrician assessment will be created with the consecutive number written down in the task description.



Description	State	Created	Due	Priority	Attachments
Laura Holmes - Obstetrician Assessment 2	✓	2/26/10 7:44 PM			
Laura Holmes - Trace FHR Monitoring 2	✓	2/26/10 7:36 PM			
Laura Holmes - Start FHR Monitoring 2	✓	2/26/10 7:36 PM			
Laura Holmes - Admission	✓	2/26/10 7:08 PM			
Laura Holmes - Exit	✓	2/26/10 7:08 PM			
Laura Holmes - Bracelet Assignment	✓	2/26/10 7:08 PM			
John Gates - Trace FHR Monitoring 1	✓	2/26/10 7:07 PM			
John Gates - Start FHR Monitoring 1	✓	2/26/10 7:07 PM			
John Gates - Stop FHR Monitoring 1	✓	2/26/10 7:07 PM			
John Gates - Obstetrician Assessment 1	✓	2/26/10 7:07 PM			
John Gates - Bracelet Assignment	✓	2/26/10 7:07 PM			
John Gates - Exit	✓	2/26/10 7:07 PM			
John Gates - Admission	✓	2/26/10 7:07 PM			

Figure 48 – Patient Tasks. After completed a 'Obstetrician Assessment' task.

A new 'Obstetrician Assessment' task is created with a consecutive number in the description for identifying it.

5. Admission

The following figures depict the two forms associated with the 'Admission task'. Once the first is completed, the second one will be opened automatically.



ACTIVE LABOUR

☐ Parous ☒ Nulliparous

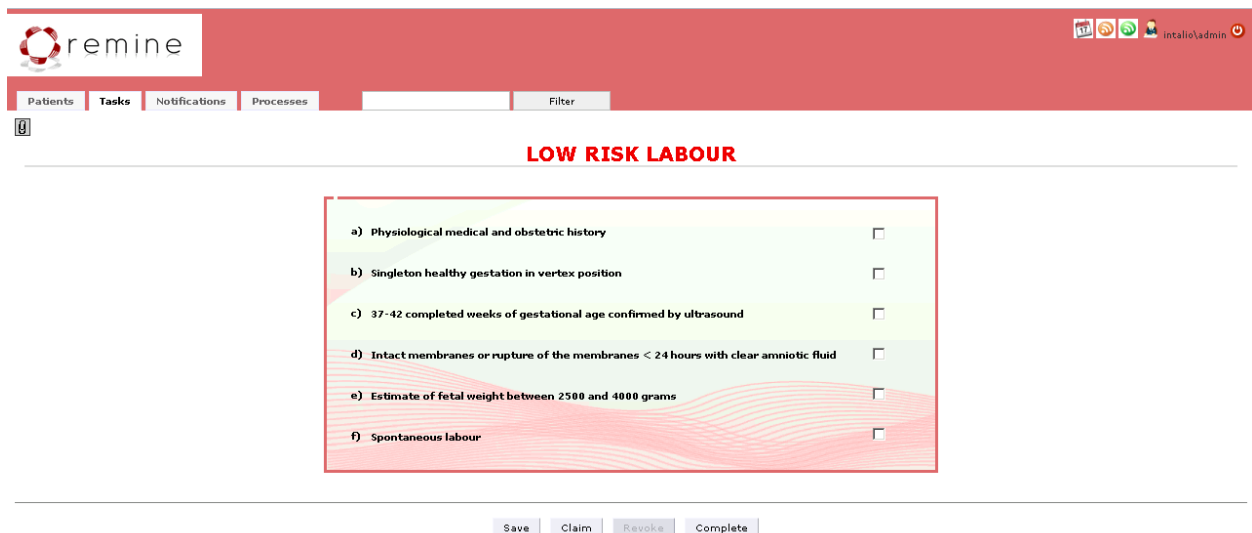
Contractions ≥ 2 in 10 minutes, regular and painful? ☐ Yes ☒ No

Cervical Length (%)

*Women with pain but no cervical changes should be re-examined after two hours

Save Claim Revoke Complete

Figure 49 – Admission 1/2. Active Labour



LOW RISK LABOUR

a) Physiological medical and obstetric history ☐

b) Singleton healthy gestation in vertex position ☐

c) 37-42 completed weeks of gestational age confirmed by ultrasound ☐

d) Intact membranes or rupture of the membranes < 24 hours with clear amniotic fluid ☐

e) Estimate of fetal weight between 2500 and 4000 grams ☐

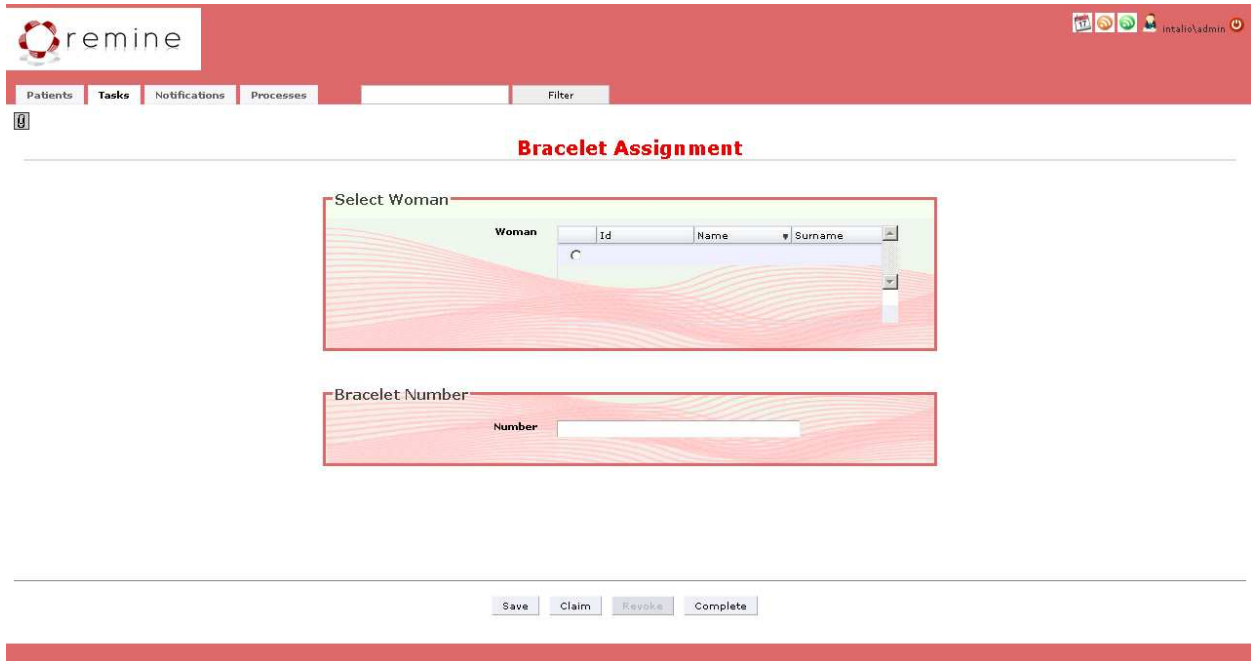
f) Spontaneous labour ☐

Save Claim Revoke Complete

Figure 50 – Admission 2/2. Low Risk Labour

6. Bracelet Assignment

The following figure represents the form to assign a bracelet number to a patient.



The screenshot shows the 'remine' web application interface. At the top, there is a navigation bar with tabs for 'Patients', 'Tasks', 'Notifications', and 'Processes'. A 'Filter' button is also present. Below the navigation bar, the title 'Bracelet Assignment' is displayed in red. The main form consists of two sections: 'Select Woman' and 'Bracelet Number'. The 'Select Woman' section contains a table with columns for 'Id', 'Name', and 'Surname'. The 'Bracelet Number' section contains a 'Number' input field. At the bottom of the form, there are four buttons: 'Save', 'Claim', 'Revoke', and 'Complete'.

Figure 51 – Bracelet Assignment

7. Exit

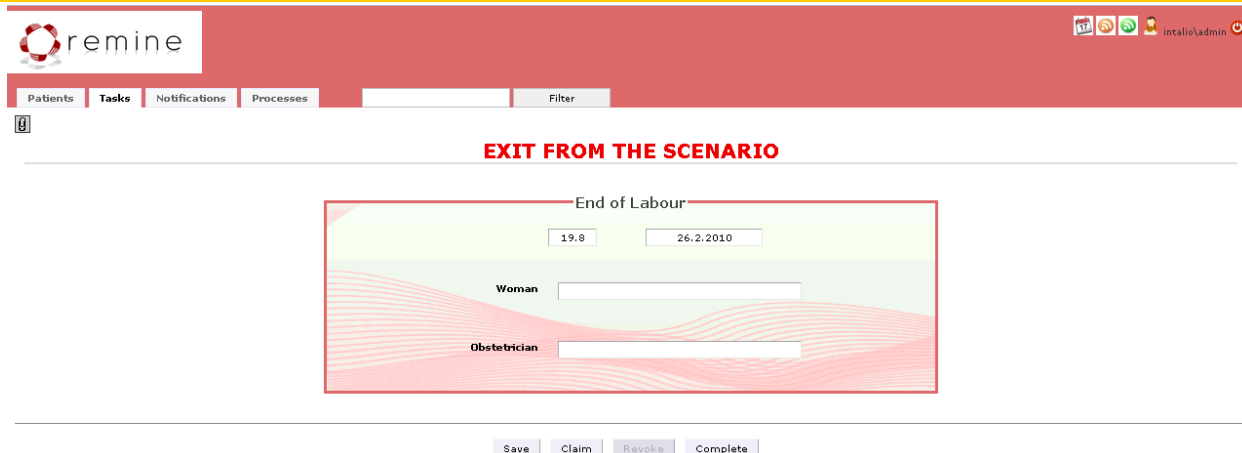


Figure 52 – Exit

The obstetrician through this form can exit from the scenario of a given patient. Therefore this task will be removed and the iterative tasks will be available but they could not create new tasks

5.2.2 Data Mapper

The more complex use of the data mapper in this process is the used to call the data base. The rest of the mappers have been explained in other documents and they are common.

Mapper for the Data Base

All the messages to the database has the same structure, it is necessary to specify in the mapper the values that we want to send in order to execute the call in the right way. The message to the database has the following elements:

- Option: the type of this value is integer and it indicates the option to execute in the database. These are the options in concordance with the value:
 - 1: Insert in the data correspondent to the execution the Patient ID and initializes the values of the number of obstetrician assessments and monitorings .
 - 2: Sending the Patient ID it returns the number of current monitorings started.
 - 3: Insert the number passed as parameter in the number of current monitorings.
 - 4: Sending the Patient ID it returns the number of current obstetrician assessments.
 - 5: Insert the number passed as parameter in the number of current obstetrician assessments.
- FormNumber: type integer. It is the value correspondent to the number of monitorings started.
- PatientID: the patient identification number.
- ObsAssessmentNumber: type integer. It is the value that indicates the number of obstetrician assessments started.

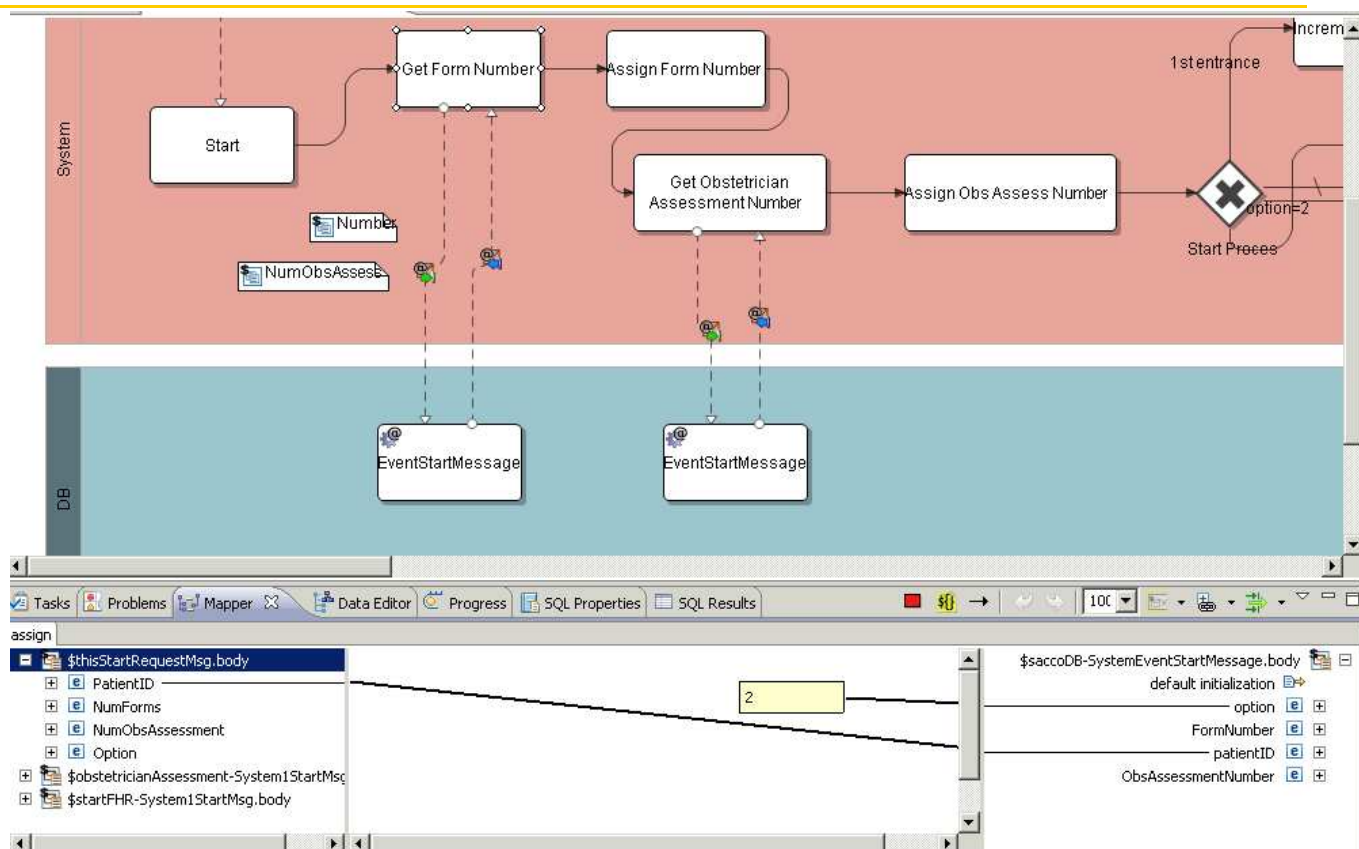


Figure 53 – Data Mapper for Database

5.3. Process Mapper for Kauhajoki

The scenario of the Kauhajoki Hospital focuses on patients hospitalized in the Acute Care for Elderly and especially in drug administration which is the most critical task in this ward. The scenario description and the steps that were used in process modeling are described in the D7.2- Pilots Test Environment Definition. The primary users of this scenario are the nurses and more specifically the admission nurse and the nurse responsible for drug preparation and administration.

For the description of the functionalities of process mapper we will present one of the scenario processes, the Admission process. The steps that were modeled are the following:

If new patient is hospitalized

- Nurse activates the ReMINE patient list update
- ReMINE creates RFID tags for 1)patient bracelet and 2)drug cups
- ReMINE associates the tags (patient –drug cup)
- Nurse applies the RFID bracelet and tags on the new patient and drug cups
- Nurse confirms the correct application of RFID tags (ReMINE GUI).

D.3.3 First Revision Data & Process model system framework

For the process execution 3 pools are created in Intalio Designer. One pool is created for the database, one for the ReMINE system and finally one for the user. The pool for the ReMINE system is defined as executable. In this pool are identified the functionalities of ReMINE that are represented as tasks and the Intalio Data Mapper is used for the mapping of the data flow between tasks.

The executable model for the Admission process is presented in the following figure:

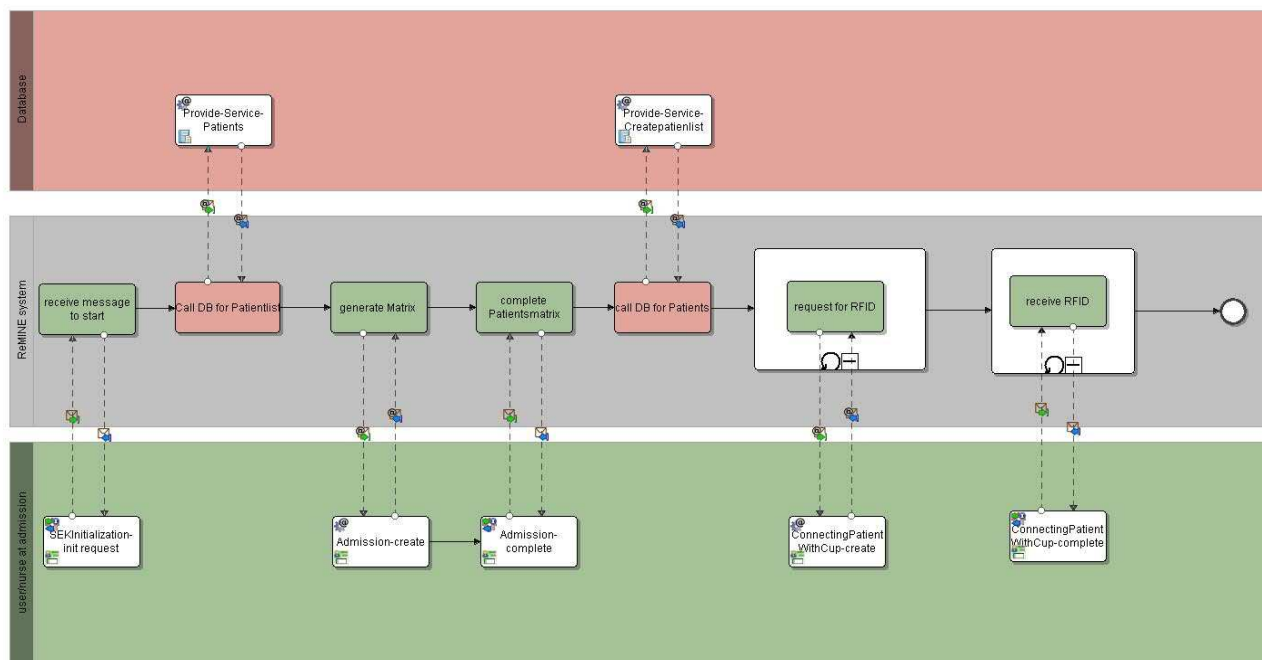


Figure 54 – Admission Executable process

The nurse will have the ability through an interface to choose the appropriate process to start running. Then ReMINE will query the database to receive the patient list with all information for generating a matrix. Afterwards, queries the database for receiving only the patient names. From there, inside ReMINE is used RFID technology and through the processes is called the appropriate web service for presenting to the user a number of the RFID for the patient and a number for the drug cup. Finally an association between the patient tag and the drug cup tag is implemented.

5.3.1 User interface

In the initial interface the end-user (nurse) by pressing the tab processes, will be able to choose the appropriate process to run. (Fig.2)

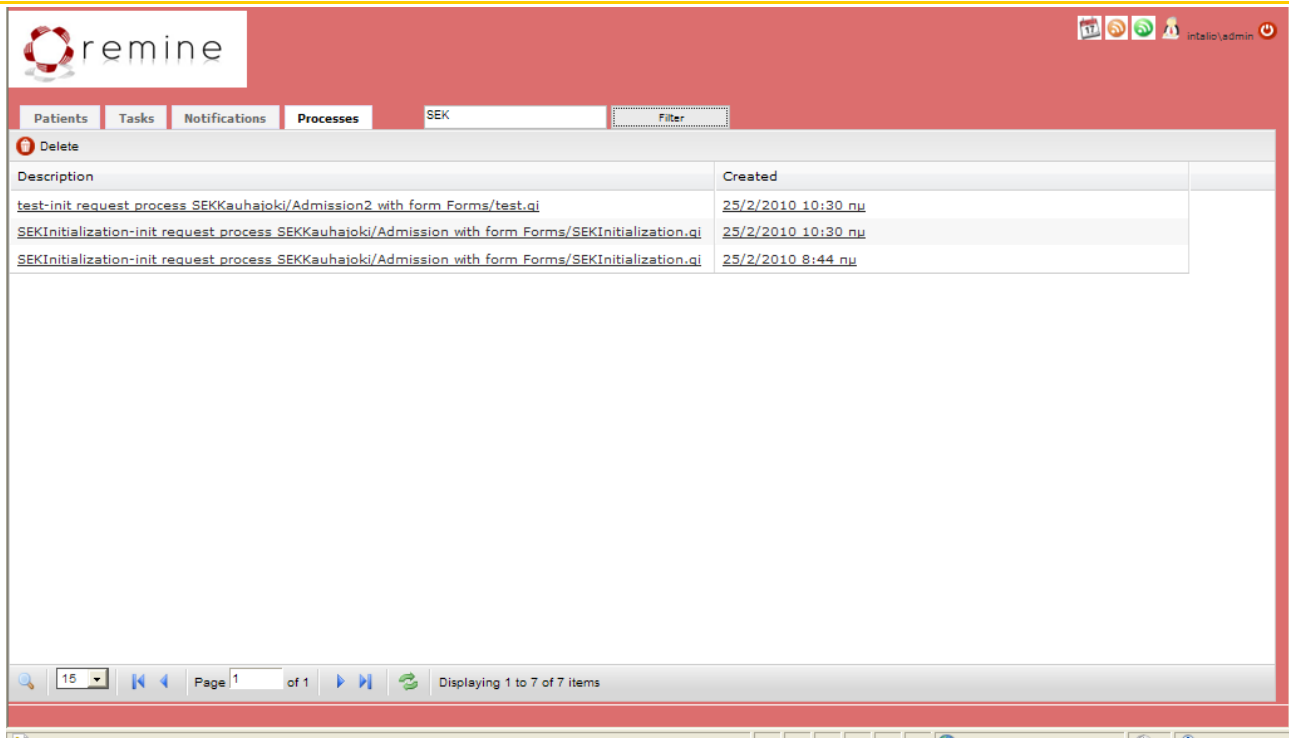


Figure 55 – Initial interface

After selecting a process to run, the user goes to the tab tasks to make the proper actions. In the next figure is presented the first task the user must choose for making a patient list generation.

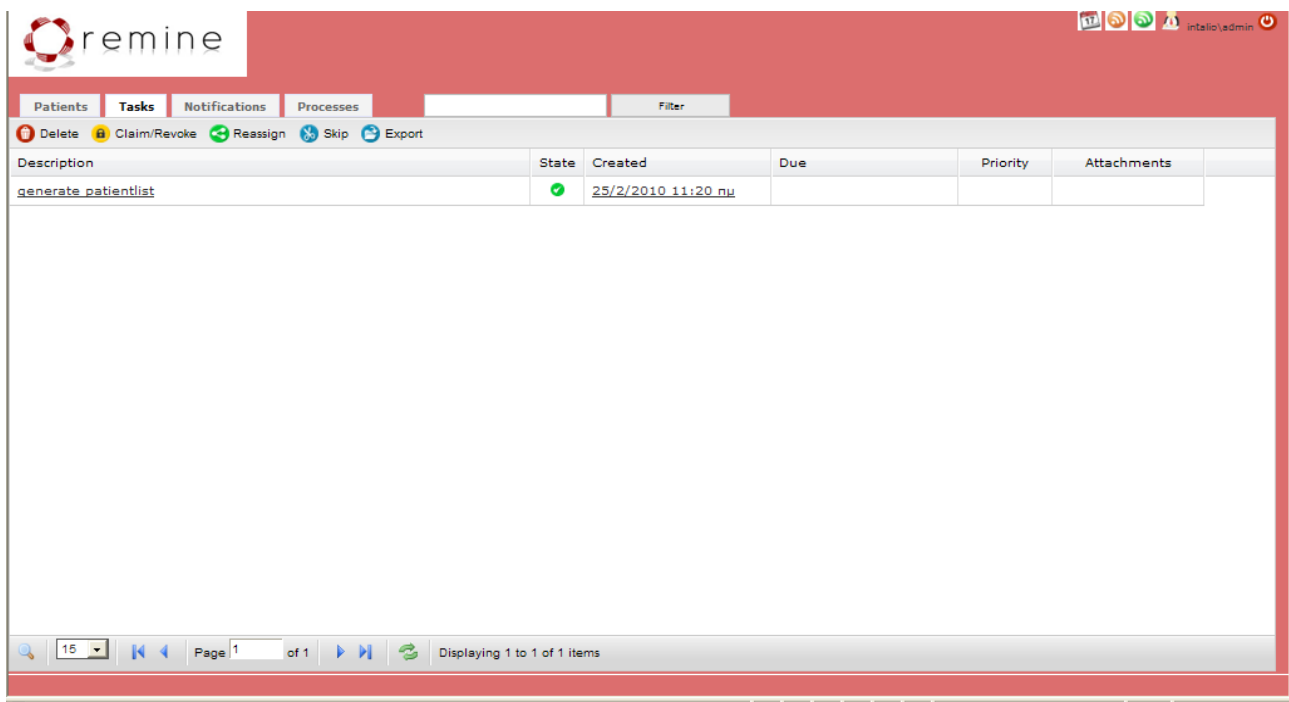
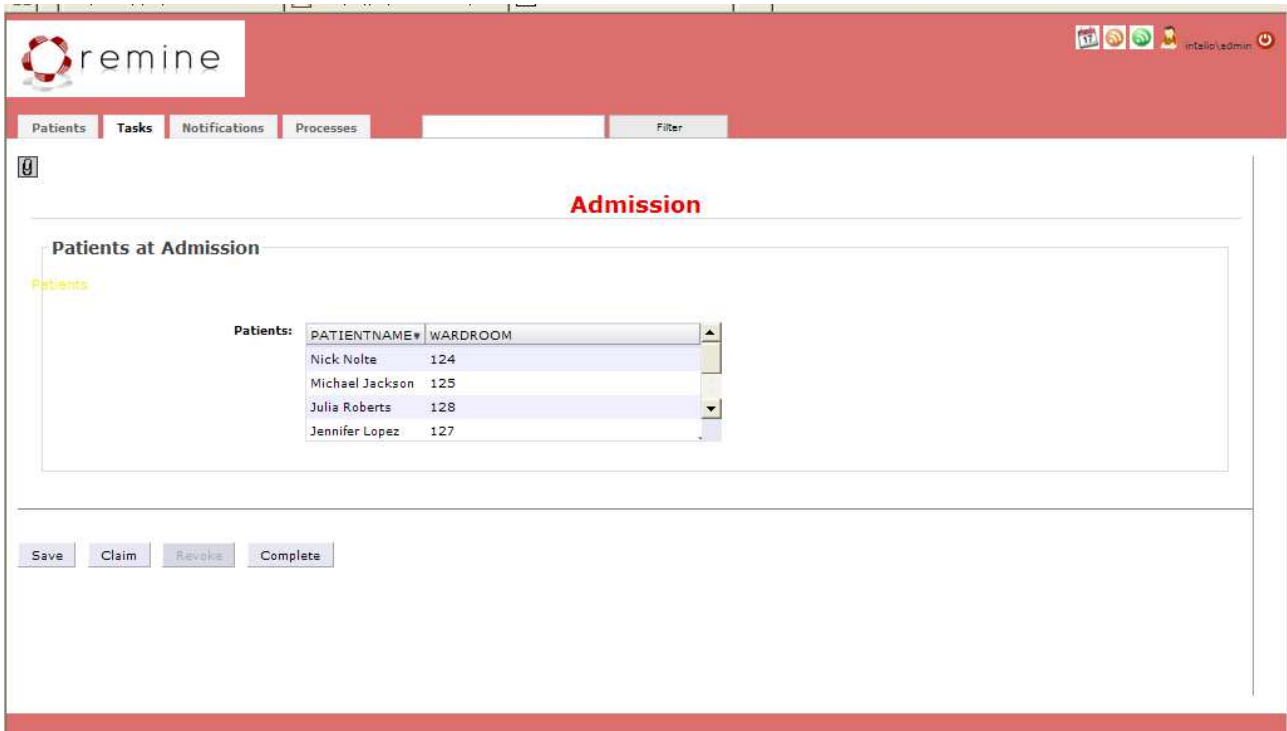


Figure 56 – Task list

The task “generate patient list” leads to the next interface where all patients are presented to the nurse. (Figure 4)

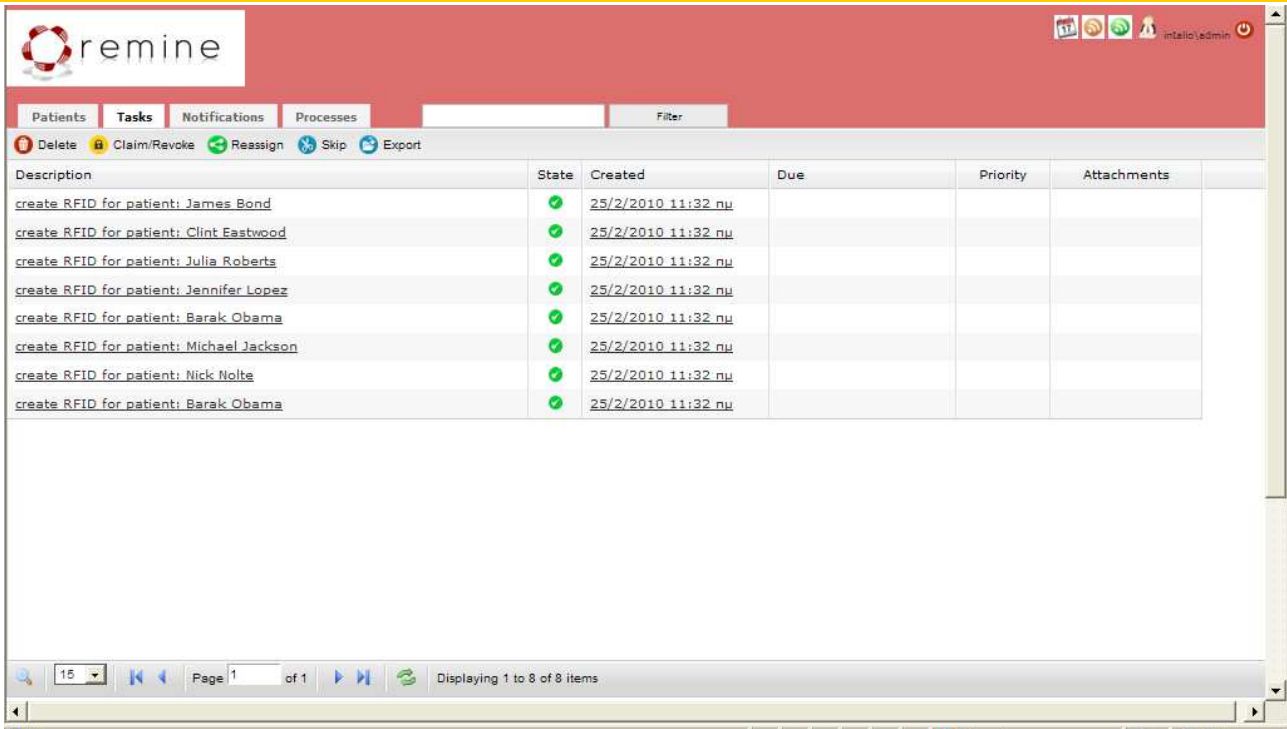


The screenshot shows the ReMINE web application interface. At the top, there is a red header bar with the ReMINE logo on the left and system icons on the right. Below the header, a navigation bar contains tabs for 'Patients', 'Tasks', 'Notifications', and 'Processes'. The 'Tasks' tab is currently selected. The main content area is titled 'Admission' in red. Below this, there is a section titled 'Patients at Admission'. A yellow highlight is placed over the word 'Patients' in this section. Underneath, a table lists four patients with their names and wardroom numbers. At the bottom of the interface, there are four buttons: 'Save', 'Claim', 'Revoke', and 'Complete'.

PATIENTNAME#	WARDROOM
Nick Nolte	124
Michael Jackson	125
Julia Roberts	128
Jennifer Lopez	127

Figure 57 – The list is generated

The user must press the button “Complete”, and then ReMINE creates for every patient, a different task.(Figure 5)

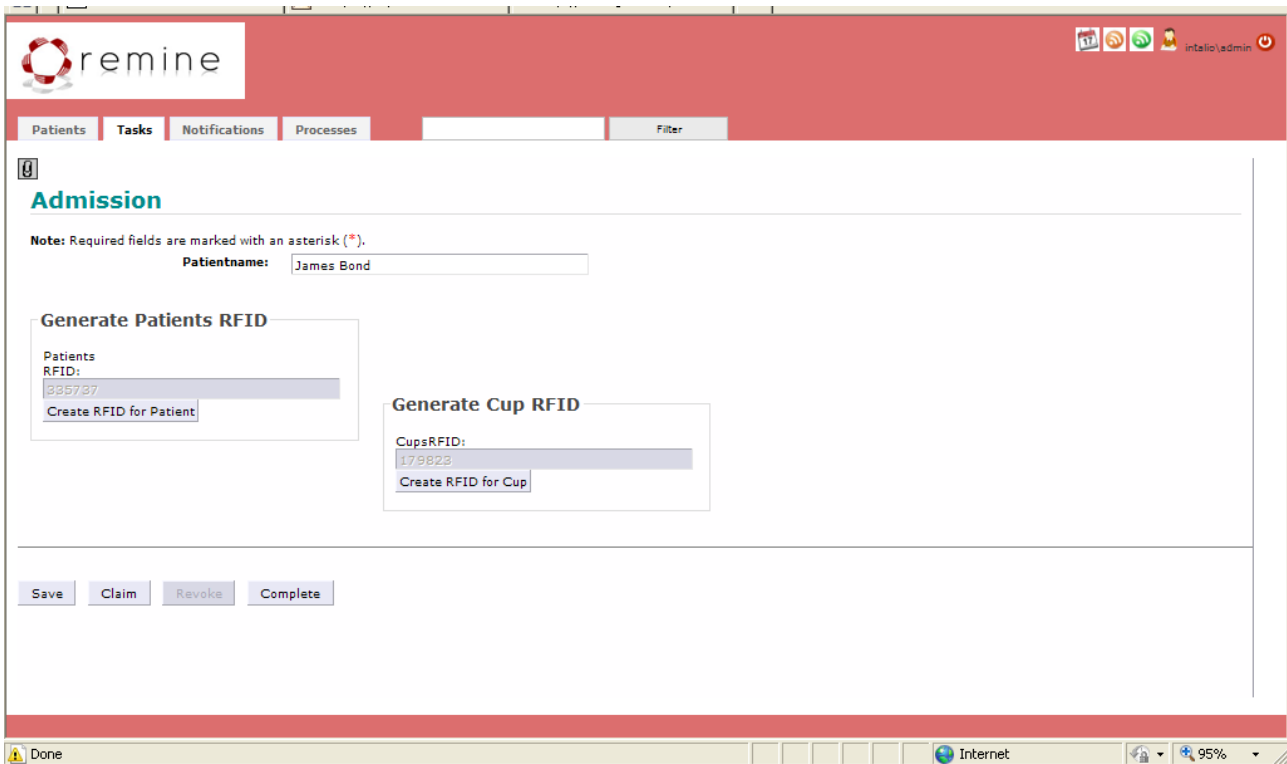


Description	State	Created	Due	Priority	Attachments
create RFID for patient: James Bond	✓	25/2/2010 11:32 nu			
create RFID for patient: Clint Eastwood	✓	25/2/2010 11:32 nu			
create RFID for patient: Julia Roberts	✓	25/2/2010 11:32 nu			
create RFID for patient: Jennifer Lopez	✓	25/2/2010 11:32 nu			
create RFID for patient: Barak Obama	✓	25/2/2010 11:32 nu			
create RFID for patient: Michael Jackson	✓	25/2/2010 11:32 nu			
create RFID for patient: Nick Nolte	✓	25/2/2010 11:32 nu			
create RFID for patient: Barak Obama	✓	25/2/2010 11:32 nu			

Page 1 of 1
Displaying 1 to 8 of 8 items

Figure 58 – Tasks to be completed for every patient

The user then can select the appropriate task for a specific patient and a new interface appears with the task that must be implemented. In this case the task is to create RFID tags for the patient and the drug cup. (Figure 6)



Admission

Note: Required fields are marked with an asterisk (*).

Patientname:

Generate Patients RFID

Patients
RFID:

[Create RFID for Patient](#)

Generate Cup RFID

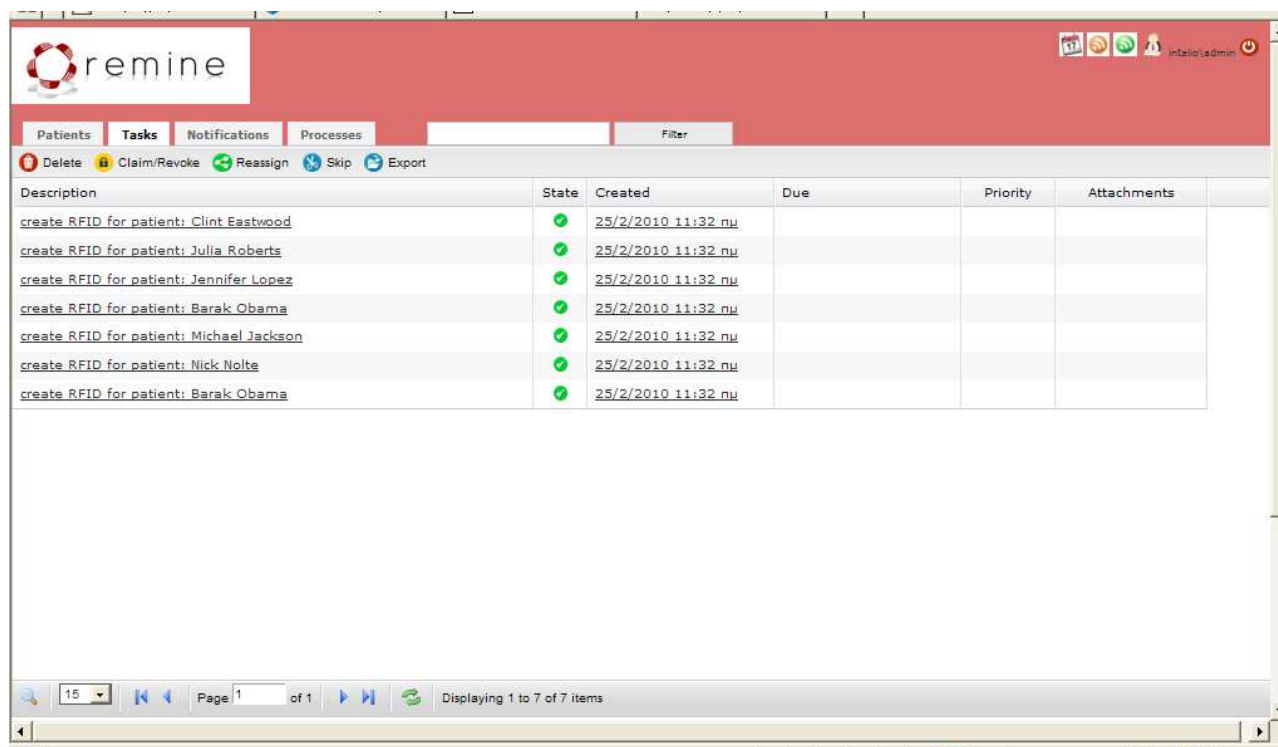
CupsRFID:

[Create RFID for Cup](#)

[Save](#) [Claim](#) [Revoke](#) [Complete](#)

Figure 59 – Creating RFID tags

In this interface the user presses the buttons “Create RFID for Patient” and “Create RFID for Cup” and automatically a serial number appears on the screen. After completing this task the user must press the button “Complete”. Then the task for this patient is deleted from the task list for not confusing the nurse with the patients that are already handled and the patients that need to be handled. In the above example we have chosen the patient “James Bond” and in the next figure it is shown that the task for this patient doesn’t exist in the list. (see also Figure 5 for comparison). Also with the button “Complete” an association inside the database of the RFID tag of the patient with the RFID tag of the drug cup is implemented with ReMINE.

**Figure 60** – The task list without the task that is completed.

5.3.2 Data Mapper

The Data Mapper creates a data flow from left to right and provides the ability to graphically define all data expressions. Mappings consist in connecting elements/operators to other elements/operators. The mapper can have multiple conditions for complex manipulations and transformations. Mapper functions have input and output area. The functions and the operators have input on left and output on the right. String values have only an output. The Data Mapper can also be used to define conditions. In the Admission Process of the Finnish pilot, the data mapper was used for manipulations and transformation in specific tasks.

In the first case that the Data Mapper is used is for the transformation in a matrix of xml data retrieved from the database through the patientmatrix.xml and the `bpel:doXsltTransform()` (from the Mapper Palette).

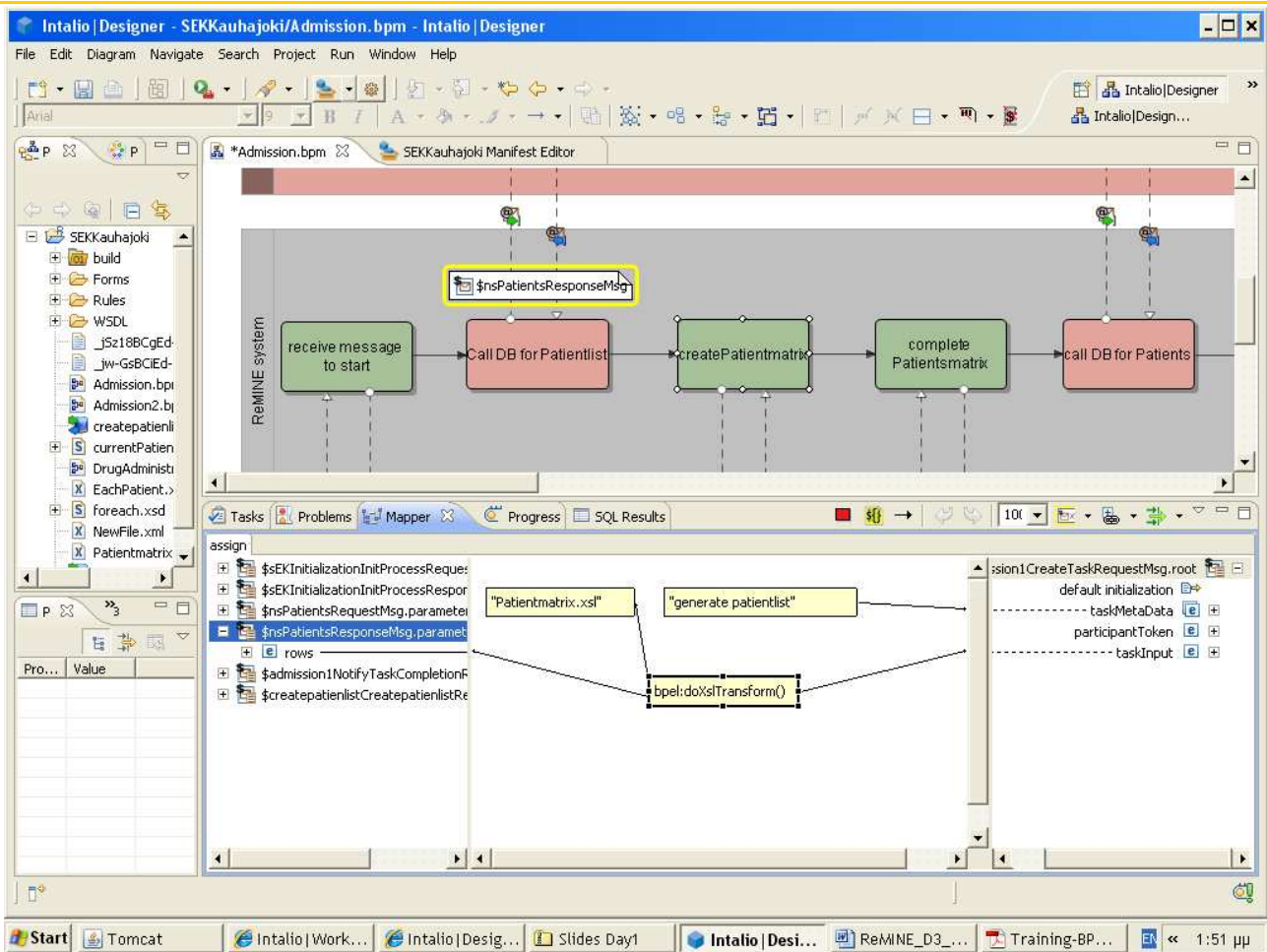


Figure 61 – Data Mapper for xml data transformation

The patientmatrix.xml file that was created is the following:

```
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0"
  xmlns:qpatients="urn:intalio.com:connectors:database:sekkauhajoki:patientservice">
  <xsl:output method="xml" />
  <xsl:template match="qpatients:row">
    <xsl:element name="row" namespace="http://www.intalio.com/gi/Admission.gi">
      <xsl:element name="PATIENTNAME"
        namespace="http://www.intalio.com/gi/Admission.gi">
        <xsl:value-of select="qpatients:PATIENTNAME" />
      </xsl:element>
      <xsl:element name="WARDROOM"
        namespace="http://www.intalio.com/gi/Admission.gi">
        <xsl:value-of select="qpatients:WARDROOM" />
      </xsl:element>
    </xsl:element>
  </xsl:template>
  <xsl:template match="qpatients:rows">
    <xsl:element name="FormModel" namespace="http://www.intalio.com/gi/Admission.gi">
      <xsl:apply-templates/>
    </xsl:element>
  </xsl:template>
</xsl:stylesheet>
```

In the second case for the Process Admission in the Finnish pilot scenario that the Data Mapper is used, is for the looping sub-processes. The looping sub-process starts with a counter from the first patient and ends when all patients are processed. In the Data Mapper the first counter and the final counter were defined. (Fig.9)

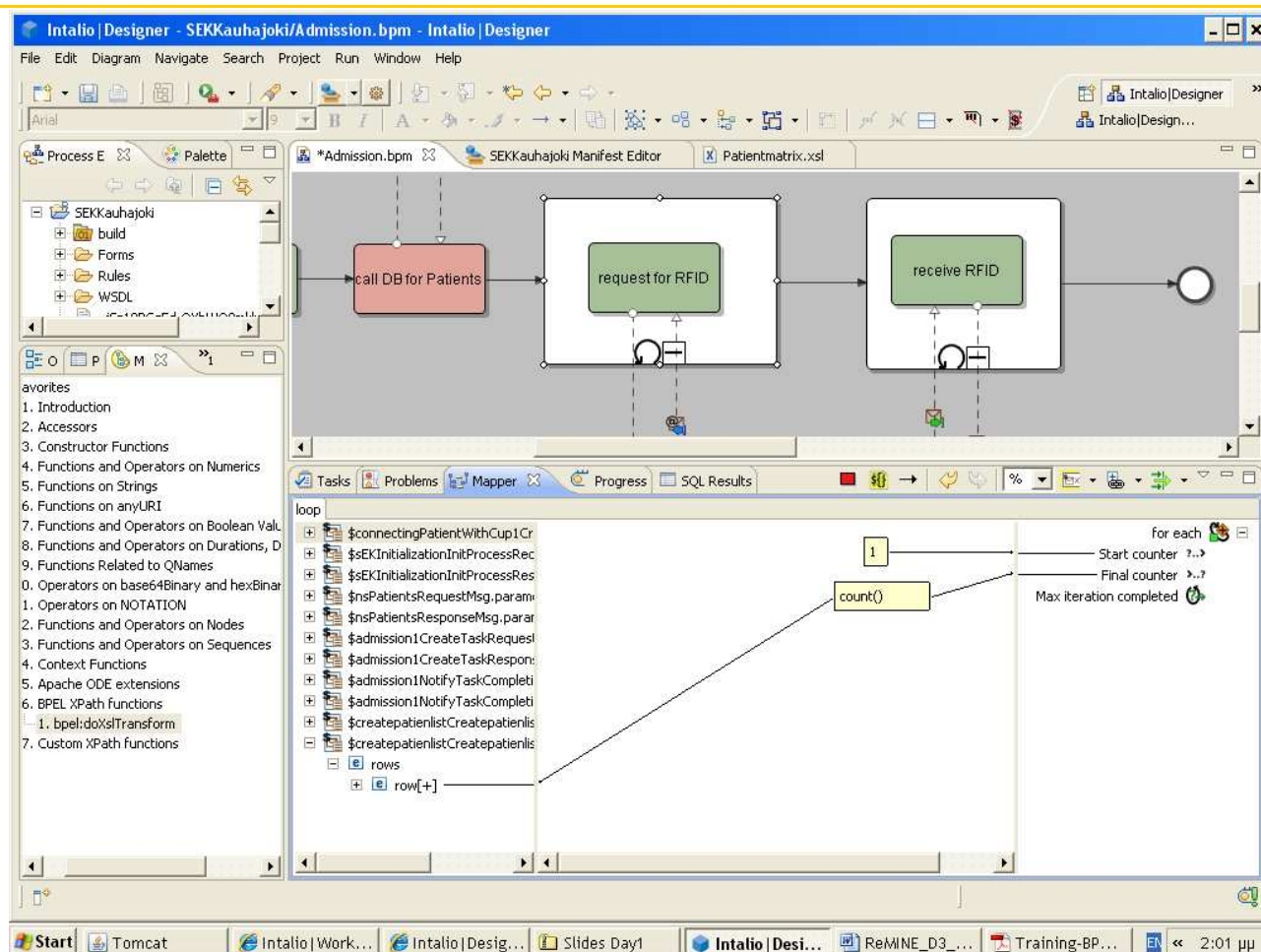


Figure 62 – Data Mapper for looping sub-process

In the third case was used for the creation of tasks for all patients. For this specific process the task is “Create RFID for patient” and for every patient this task is presented in ReMINE task list. (see Fig.5). In the next screenshot is shown the Data Mapper for task creation.

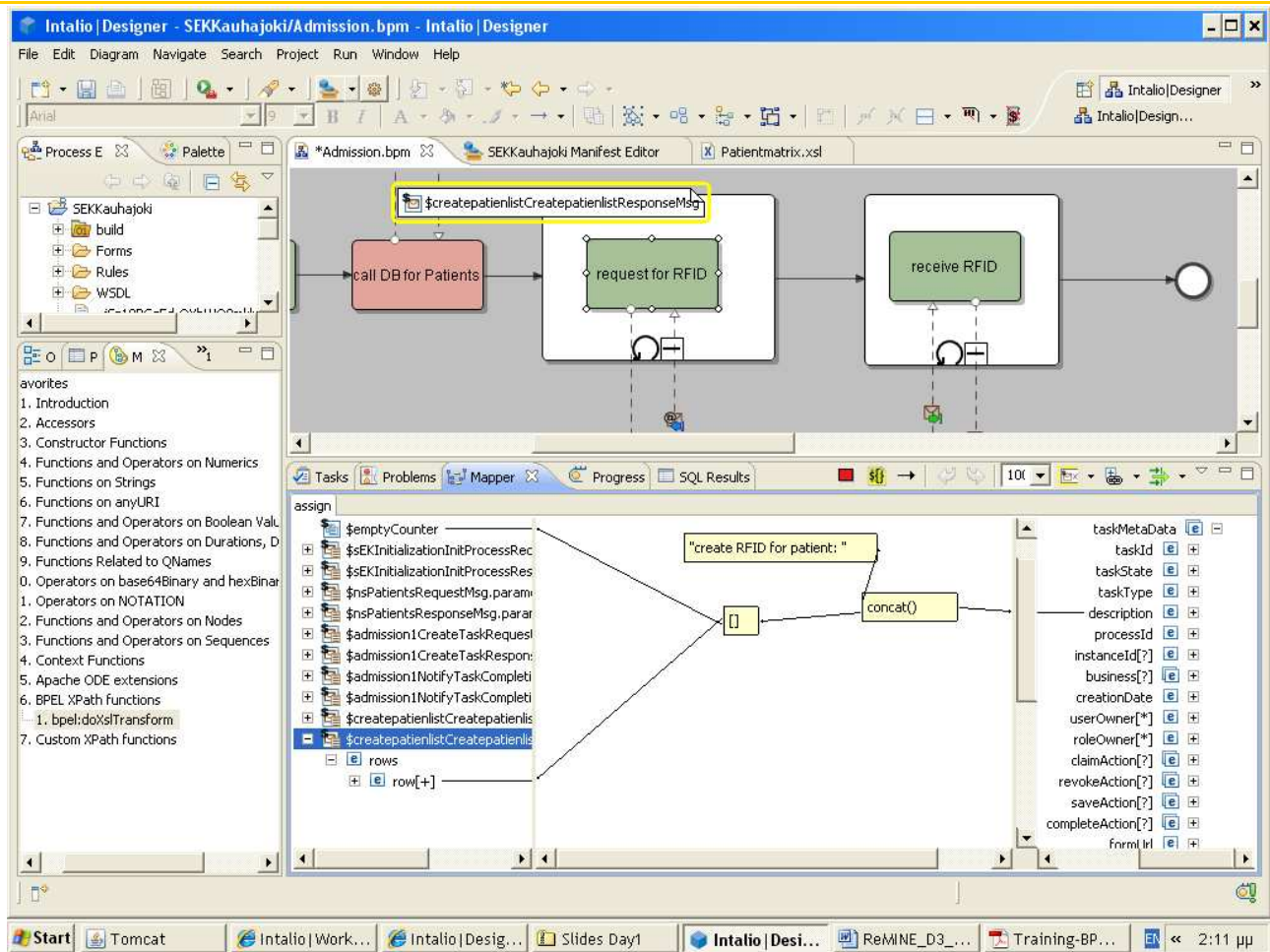


Figure 63 – Data Mapper for task creation

5.3.3 RFID integration

As described in paragraph 5.2.1 an interface where the user can create RFID tags with ReMINE (Fig.6) has been implemented with the use of XML Mapping Utility inside the General Interface of Intalio.

The XML Mapping Utility is used to map the GUI component “createRFIDpatient” and “createRFIDcup” with a web service.

Following steps:

- **Create mapping rules file** : Mapping rules files are XML files that define mappings between application objects and data elements. The XML Mapping Utility provides a drag-and-drop interface for creating mappings.

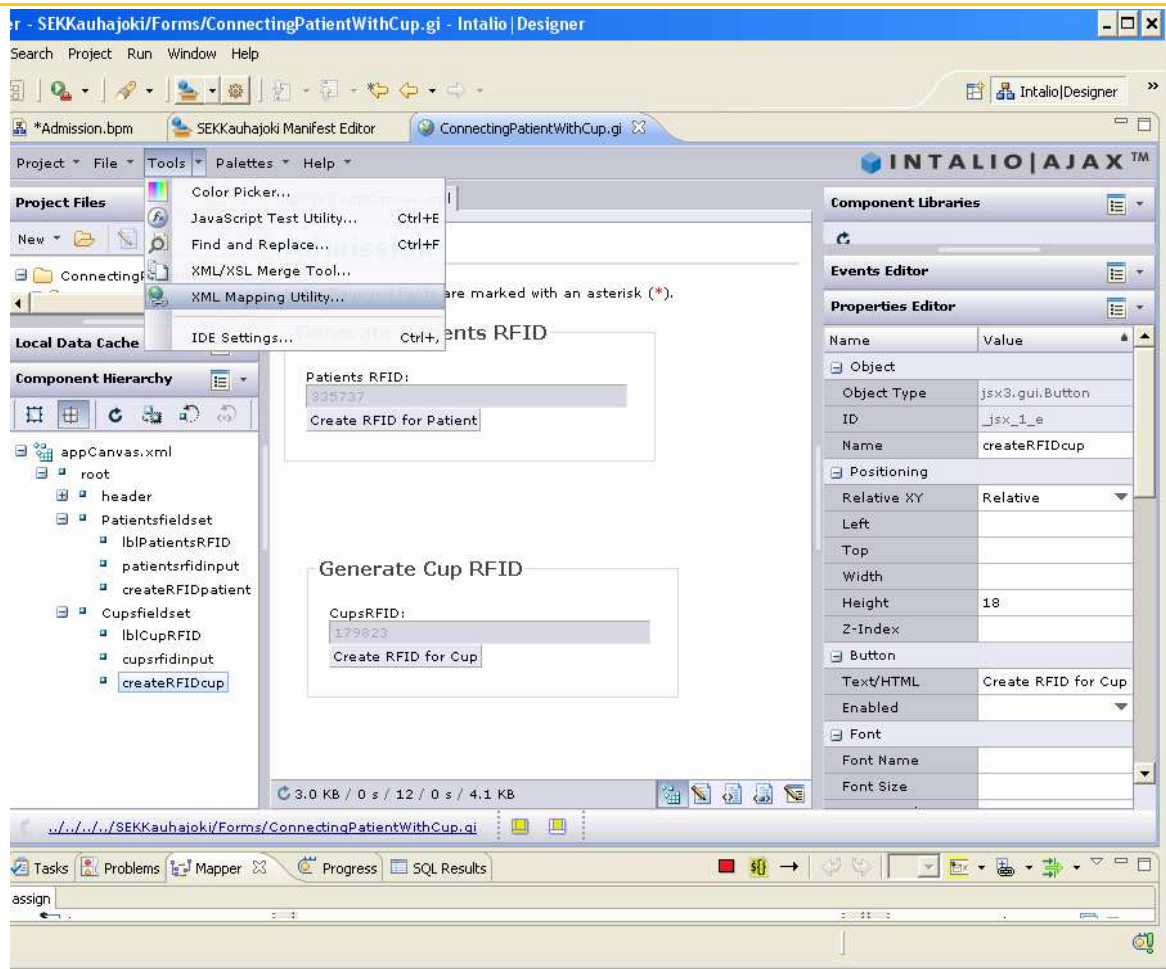


Figure 64 – Creation of xml mapping utility

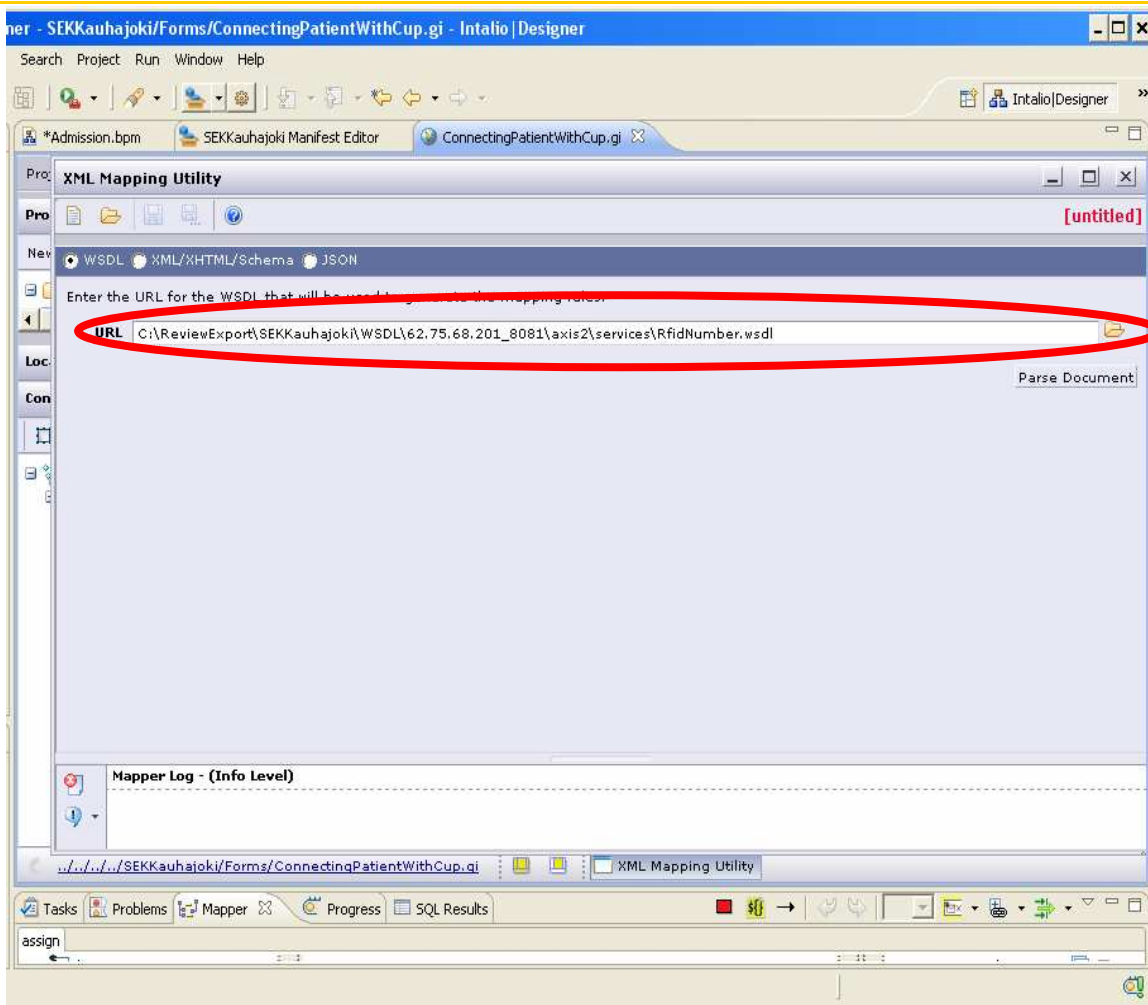


Figure 65 – Web service' s URL insertion

By selecting “Parse Document” button, the Rules Tree panel appears as it is shown in the next figure:

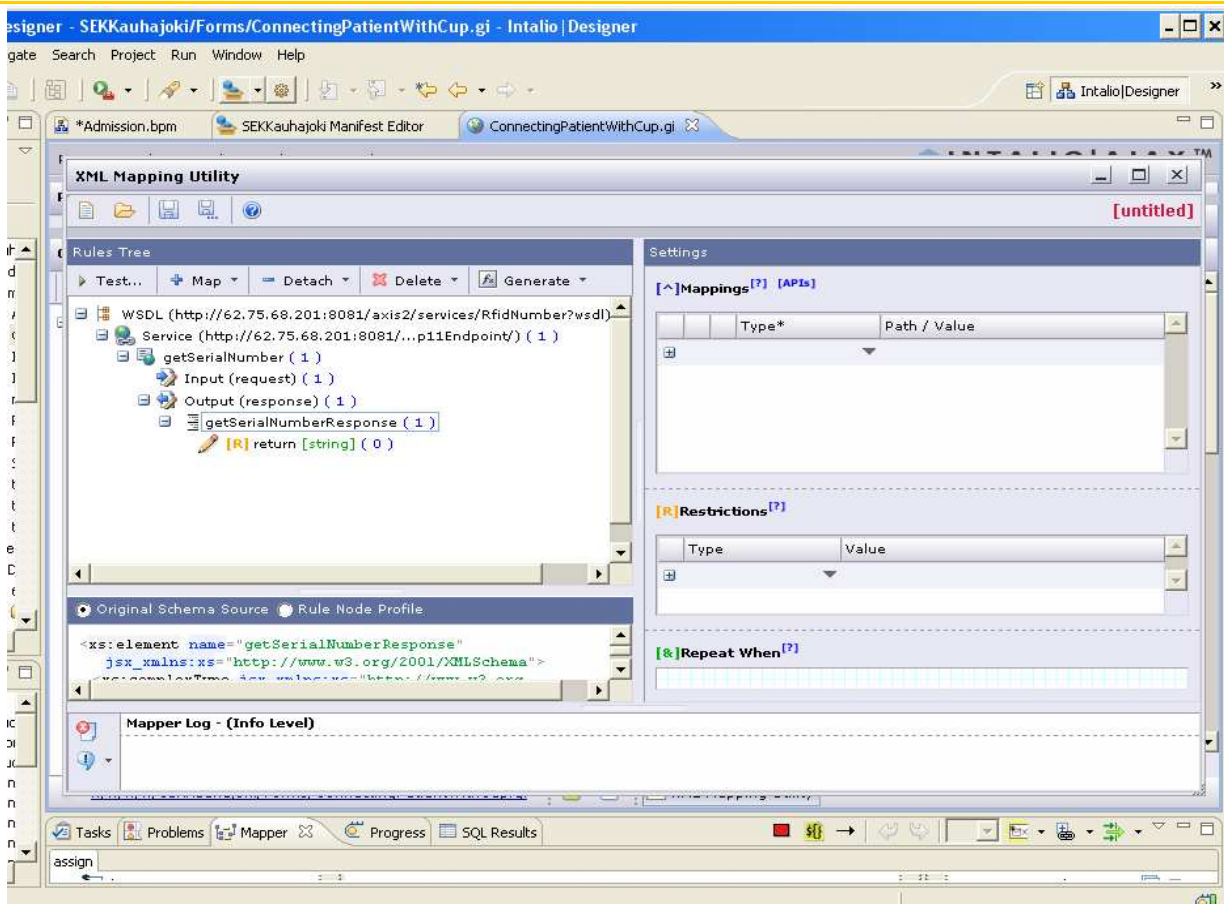


Figure 66 – Rules tree panel

- **Defining mappings:** In this step, mappings between the GUI components of the “ConnectingPatientWithCup.gi” and the data exposed by the web service are defined. In the application is used the getSerialNumberResponse operation for the rule file.
- **Defining Mappings for Input Data.** To define mappings between the GUI components and the input (request) data the next steps are followed :
 1. Select the getSerialNumberResponse operation in the Rules Tree panel.
 2. Create a mapping between the “patientsrfidinput” component and the return rule as follows: Drag the patientsrfidinput component from the Component Hierarchy palette to the return rule in the Rules Tree panel. The following mapping is created:

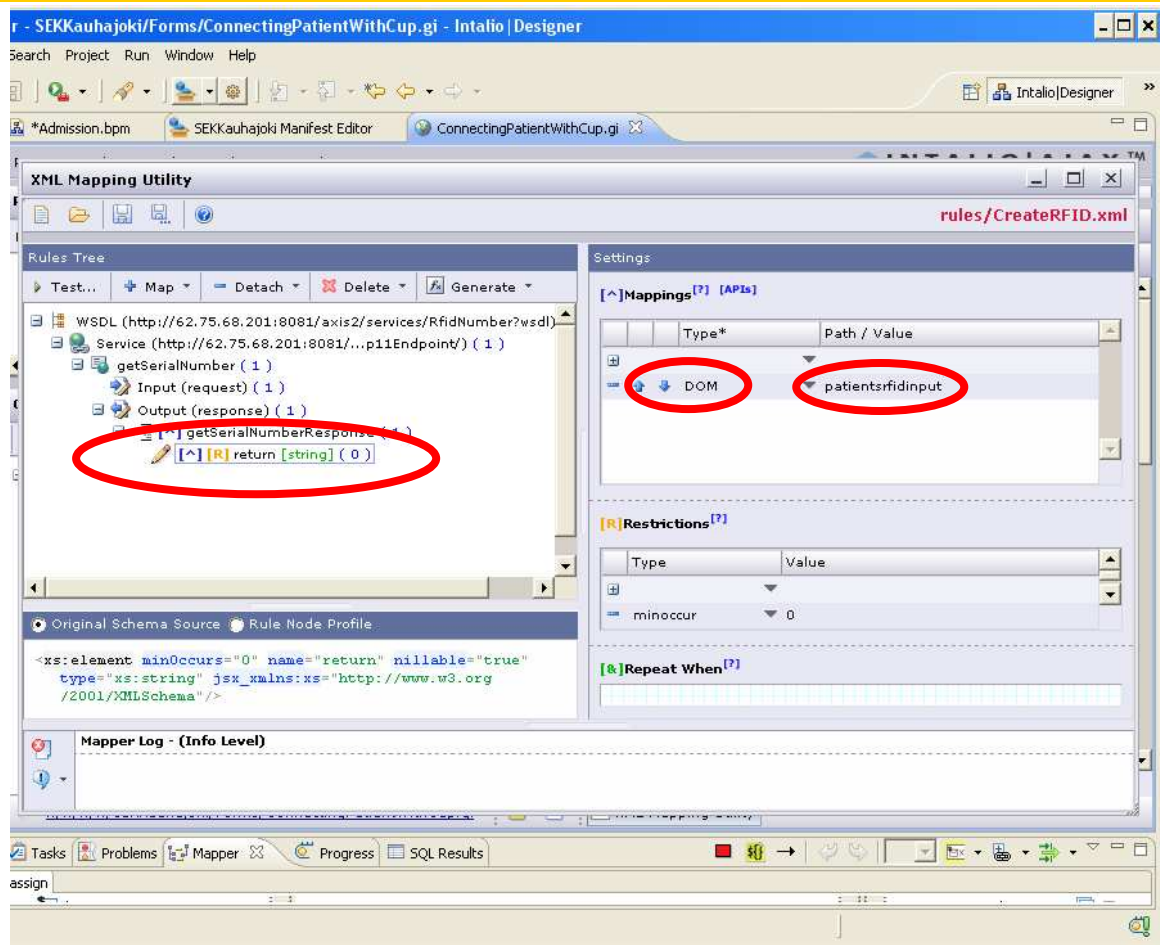


Figure 67 – Mapping for input data

The Mappings table is on the right side and the Type is DOM (Document Object Model), representing a mapping between a GUI component and a rule in the rules tree.

The value is patientsrfidinput which is also the name of the GUI component.

- **Generating Function Code**

A rules file defines the interaction between the application and the data service. Once the rules file is complete, the application needs to invoke the service. The XML Mapping Utility generates the function code that invokes the service, which is added to the project in an included JavaScript file. It is configured the “createRFIDpatient” button to execute the JavaScript function eg. service.callgetSerialNumber();.

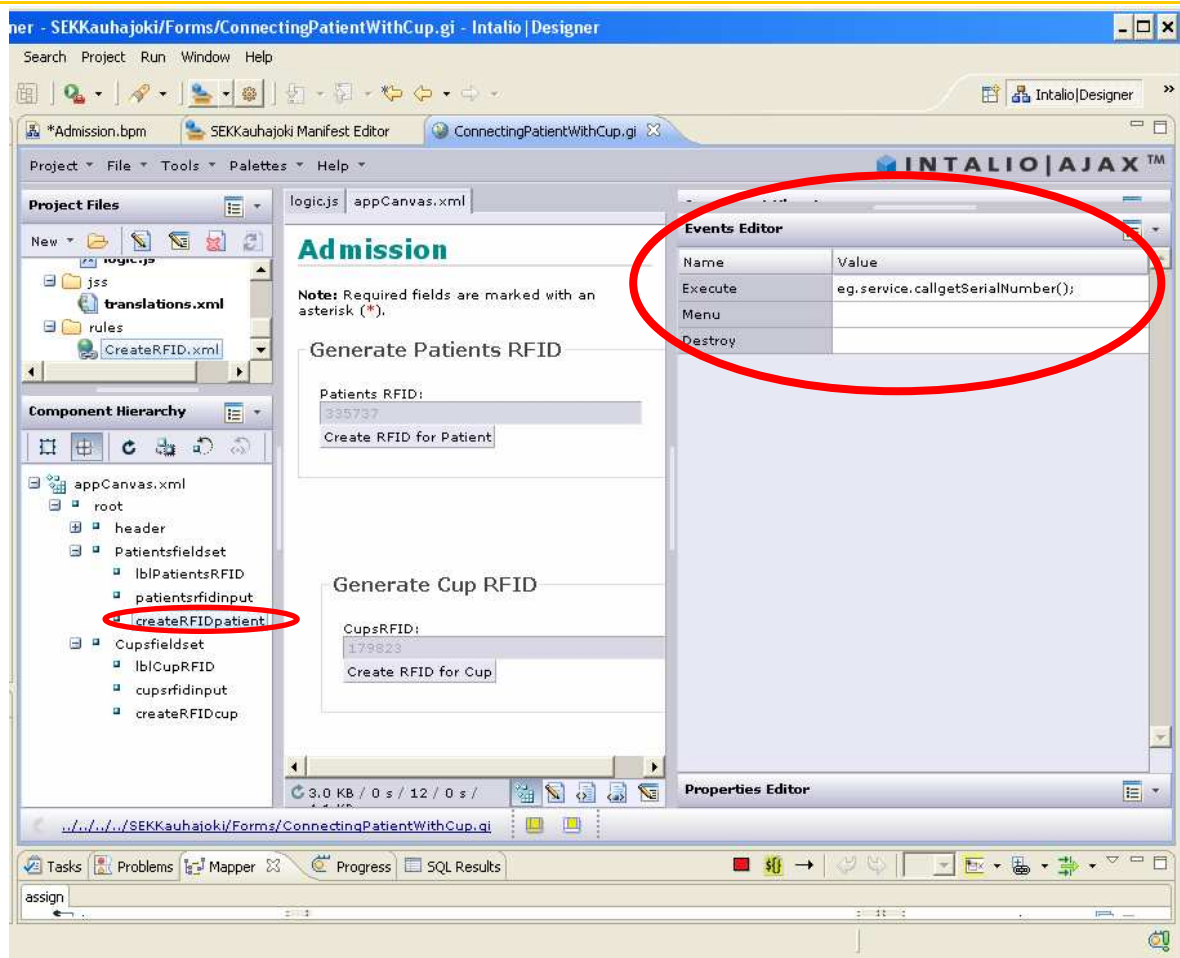


Figure 68 – Event Editor

- By clicking the “Generate” button, the function code that invokes the web service is then generated:

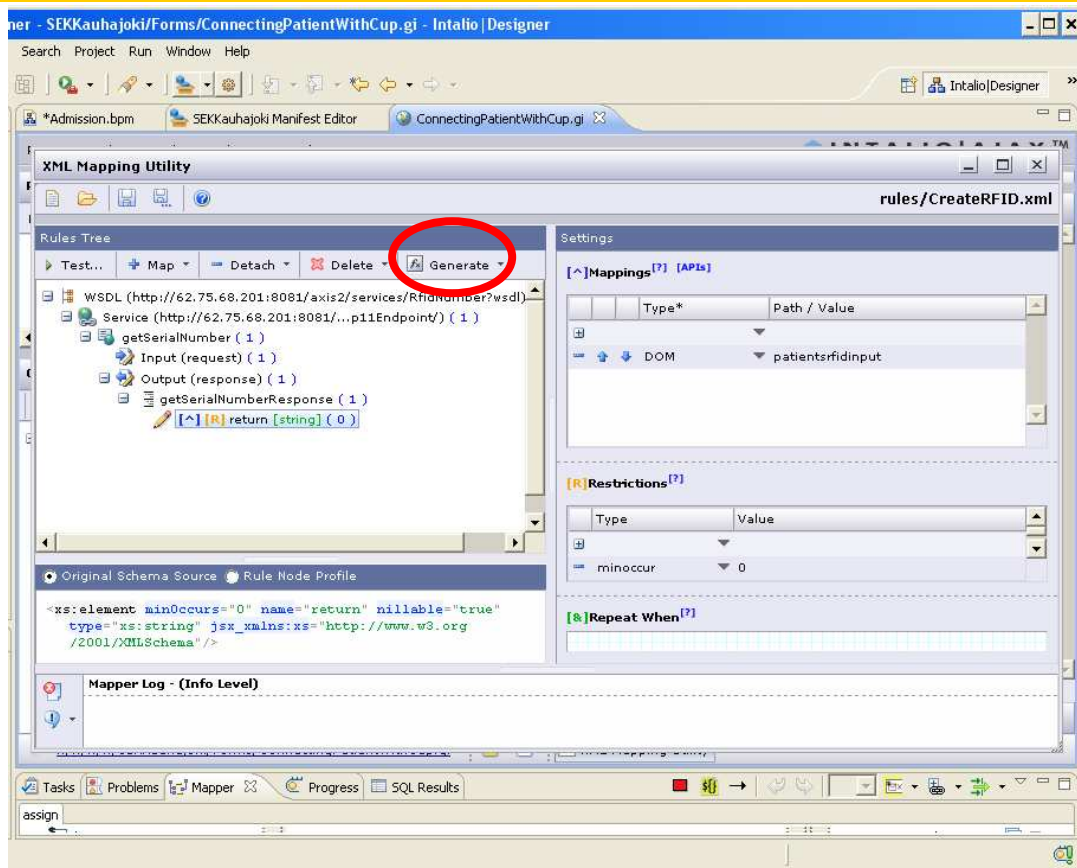


Figure 69 – Button “Generate”

The generated function code is the following:

```

jsx3.lang.Package.definePackage( "eg.service",           //the full name of the package to
create function(service) {                               //name the argument of this function
    //call this method to begin the service call (eg.service.callgetSerialNumber());
    service.callgetSerialNumber = function() {
        var objService = ConnectingPatientWithCup.loadResource("CreateRFID_xml");
        objService.setOperation("getSerialNumber");
        //subscribe
        objService.subscribe(jsx3.net.Service.ON_SUCCESS, service.ongetSerialNumberSuccess);
        objService.subscribe(jsx3.net.Service.ON_ERROR, service.ongetSerialNumberError);
        objService.subscribe(jsx3.net.Service.ON_INVALID, service.ongetSerialNumberInvalid);

        //PERFORMANCE ENHANCEMENT: uncomment the following line of code to use XSLT to convert the
server response to CDF (refer to the API docs for jsx3.net.Service.compile for implementation
details)
        //objService.compile();
        //call the service
        objService.doCall();
    };
    service.ongetSerialNumberSuccess = function(objEvent) {
        //var responseXML = objEvent.target.getInboundDocument();
        objEvent.target.getServer().alert("Success","The service call was successful.");
    };
    service.ongetSerialNumberError = function(objEvent) {
        var myStatus = objEvent.target.getRequest().getStatus();
        objEvent.target.getServer().alert("Error","The service call failed. The HTTP Status code is: "
+ myStatus);
    };
    service.ongetSerialNumberInvalid = function(objEvent) {
        objEvent.target.getServer().alert("Invalid","The following message node just failed
validation:\n\n" + objEvent.message);
    };
}

```



```
};
```

This function code is added into the logic.js tab

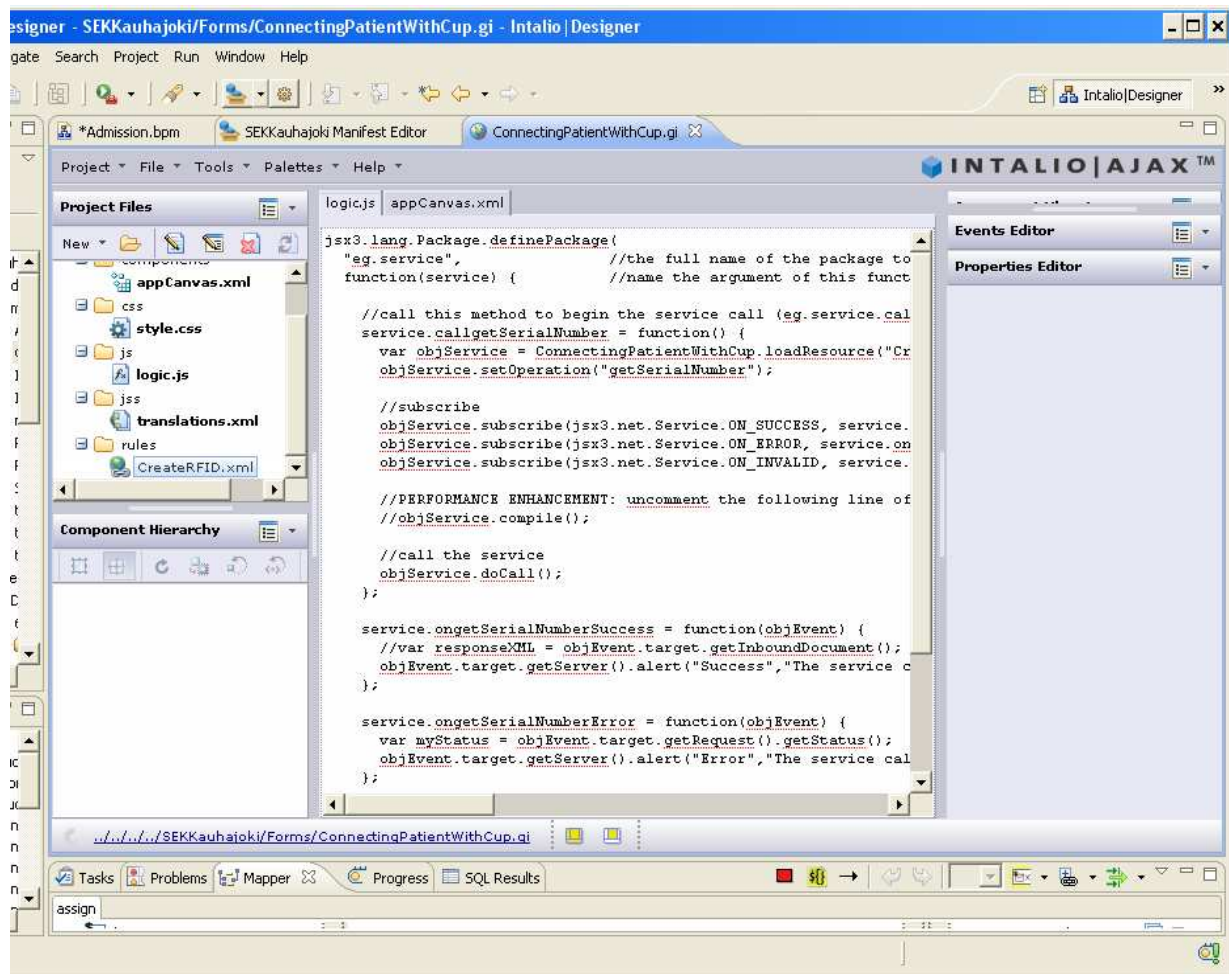


Figure 70 – logic.js tab

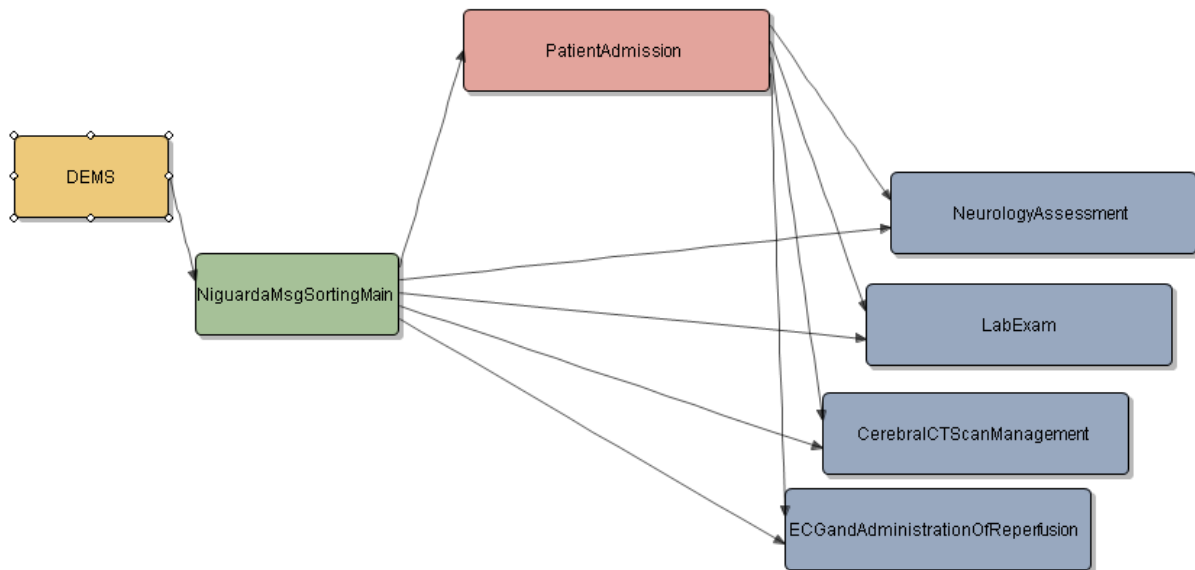
5.4. Process Mapper for Niguarda

5.4.1 Introduction

This section describes the purpose and realization of the model for Niguarda pilot. This model provides monitoring of hospital processes. In principle it checks if each step of a hospital process is executed in the right order and at the right time, if not this model notify to the responsible person(s) that the hospital process have not been executed right.

5.4.2 Overview

The model contains 7 parts(diagrams): NiguardaMsgSortingMain, PatientAdmission, ReAssessment, NeurologyAssessment, LabExam, CerebralCTScanManagement and ECGandAdministrationOfReperfusion. The last four diagrams do monitoring activities over hospital processes. "NiguardaMsgSortingMain" diagram just receive messages from DEMS and resend them to the proper instances of other diagrams.



Instance over “PatientAdmission” diagram is triggered by ‘Triage’ message. If the content of Triage message fulfills some conditions (we need more detailed description of these conditions) the diagram triggers all 4 monitoring activity diagrams.

Then monitoring activity diagrams will just wait for other messages from “NiguardaMsgSortingMain” diagram. It means that for every incoming message from “DEMS” there will be exactly one instance of “NiguardaMsgSortingMain” diagram.

ReAssessment diagram is triggered by receiving “ReAssessment msg”. If the content of the msg fulfills the conditions the monitoring activity of this patient ends. In consequence the process over this diagram will kill all monitoring processes of the current patient.

5.4.3 Correlation

Correlation provides that all received messages go to the correct business process instances. The solution that is used in niguarda model is not definitive. For now, Niguarda model uses only one correlation set which has one property. This property is set on “IDs” parameter of incoming messages. Every kind of message that Niguarda model can receive has “IDs” parameter. So if there is a new incoming message with “IDs” parameter, it is delivered to monitoring activity instance which were started with the message with the same “IDs” value.

This solution is good for testing but not for using. For example, there should be more than one Lab examination for one patient and this solution enables only one.

5.4.4 “NiguardaMsgSortingMain” model

There is no scenario for this model. It handles all incoming messages and forwards them to the correct instances of monitoring processes. It is possible to send messages directly to models, but we decided to receive every incoming message in the model NiguardaMsgSortingMain, because then you need only one wsdl for sending messages to Niguarda model. Also if we would like to set some business activity

monitoring (BAM) over incoming messages, it would be better to set them in one model, no to do the same thing for 5 models.

5.4.5 “PatientAdmission” model

The scenario for patient admission is very simple. Triage nurse does an assessment of the patient that has arrived at the A&E. Triage nurse alerts a neurologist consultant and fills the electronic A&E report with the patient data and the priority code. If symptoms and priority code match with the ones defined by the stroke protocol, ReMINE (model) starts the countdown for administration of the reperfusion treatment.

So this model is also very simple. Process (ReMINE) is triggered by receiving of “Triage” message. After that process analyze the data from “Triage” msg to find out if the patient will need some examination. If the patient will need some examinations, the process will trigger processes for monitoring hospital activities. Triggering of monitoring processes is made by using “Triage” msg to invoke monitoring processes over niguarda model.

“Triage msg” has these elements:

- ID’s (Patient Id + Visit Id)
- Patient demographics
- Priority code
- Diagnosis allowing to detect the patient as “yellow stroke”
- Clinical information collected for the patient
- Time of the end of triage
- Author of the triage

More detailed description of “Triage msg” you can find in document “ReMINE_D7.2_NoemaLife_WP7_V1.0_Final.pdf”.

5.4.6 “NeurologyAssessment” model

The whole idea of this part of scenario is that, neurology consultant has 30 minutes from start of triage to do an assessment of the patient. If the neurology consultant did not manage to do the assessment model (ReMINE) should send a message to the responsible person.

Basically this model monitors if there is a report with the results of neurology assessment on time. If it is ready monitoring process just ends, if it is not ready, ReMINE sends an alert to the responsible person. Expiration time (in scenario 30 minutes) should be in model dynamic. It means it should be loaded from external source. This external source is the Stroke protocol, but for now is not clear how we will get data

from the Stroke protocol so it was decided that niguarda model should simulate this for now by loading data from WS. Now it is represented by the task “Get Date from DB” in the model.

“NeuroAssessment msg” has these elements:

- ID's (Patient Id + Visit Id)
- Patient demographics
- Clinical information collected for the patient
- Clinical information related to the executed assessment
- Time of assessment
- Doctor in charge of patient

More detailed description of “Triage msg” you can find in document “ReMINE_D7.2_NoemaLife_WP7_V1.0_Final.pdf”.

5.4.7 “CerebralCTScanManagement” model

This part of the scenario is very similar to the neurology assessment part. Neurology consultant has 30 minutes (according to the scenario) from the triage start to the order cerebral CT scan. If the CT scan order is not registered in that time ReMINE alerts to the neurology consultant or to a different person. Then the radiologist has another 30 minutes (60 minutes from triage start) to evaluate CT scan or to send the image to RIS. If it is not done, ReMINE should include this case into the monthly report for hospital risk manager.

So the principle of this model is the same as in the model above. The only difference is that this model has to monitor 2 instead of 1 incoming messages. After triggering this monitoring process, it waits on “Rad Order” message. If this message did not arrive on time the process will alert a responsible person and wait for the message to come. If the message comes on time the model starts to monitor the incoming second message “ImagelsReady”. We know that both of these messages will come in this order. There are same rules for message “ImagelsReady” as it was for “Rad Order”. So if the message “ImagelsReady” would not come at time, the model will send alert message to responsible person. Otherwise the monitoring process ends. A message should be added to the monthly report for hospital risk manager, but this has not been done yet, so the model just notifies to the responsible person.

“RadOrder msg” has these elements:

- ID's (Patient Id + Visit Id)
- Patient demographics

- Order ID
- Order date and time
- Ordered examination
- Doctor in charge of patient

“ImageReady msg” has these elements:

- ID’s
- Patient demographics
- Order ID
- Order status
- Ordered examination
- Examination status
- Examination Execution date and time

More detailed description of these messages you can find it in document “ReMINE_D7.2_NoemaLife_WP7_V1.0_Final.pdf”.

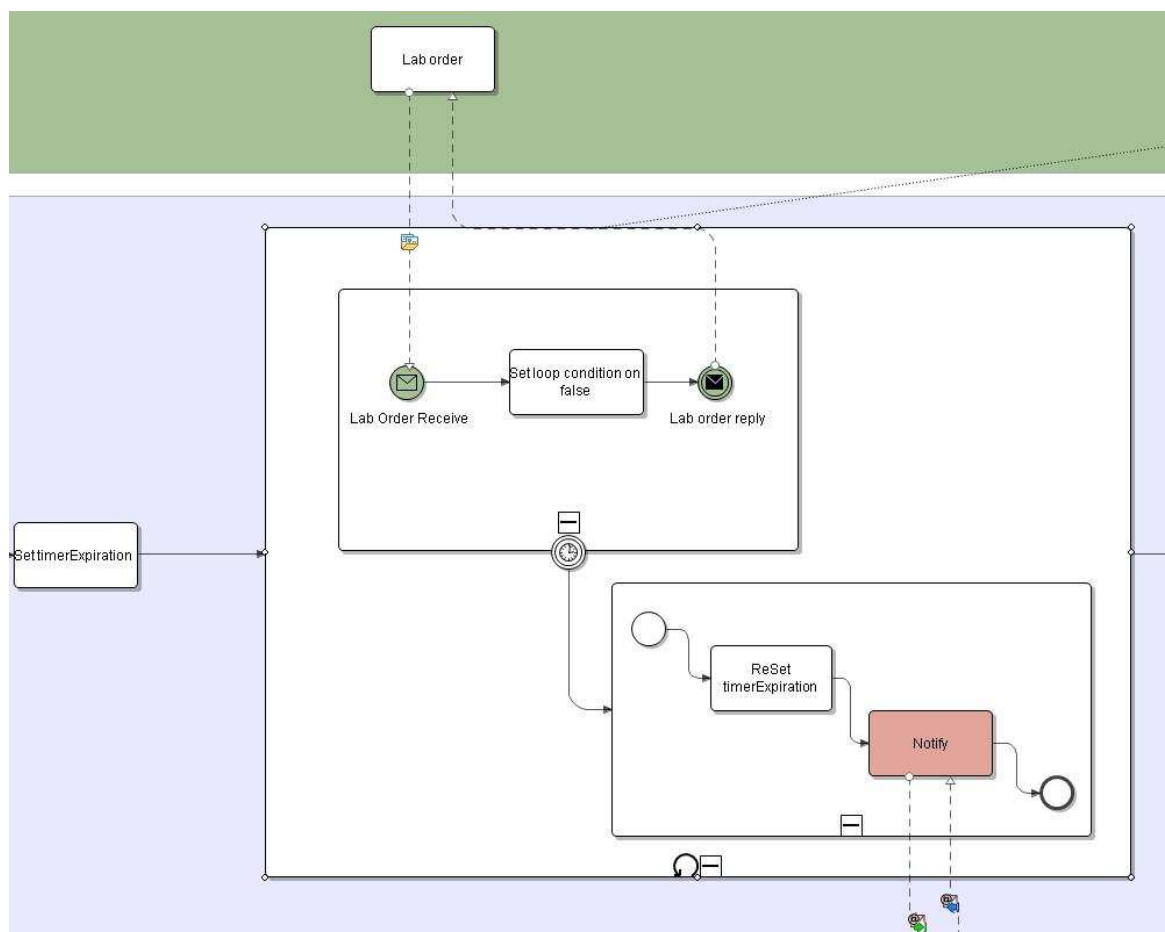
5.4.8 “LabExam” model

If the result of the triage says that it is necessary to make the patient blood exams, the neurology consultant has a certain time (according to the scenario 30 min from the triage start) to order the blood exams. If any laboratory exam order is not registered our process should alert to some responsible person. In case that the blood exam order was registered, there should be a response in some amount of time (set by the hospital risk manager) from the laboratory advising that the blood samples were checked-in. If the blood samples are not checked-in on time, the process should alert to some responsible person and wait for the “LabCheckin” message. After the “LabCheckin” message arrives, there are 45 minutes (according to scenario) from the triage start to get results of the blood examination. If the “LabResults” message did not arrive on time, the process should alert to some responsible person and wait again for “LabResults”. If “LabResults” arrives on time, the monitoring process should end. In the current version of Niguarda model waiting time has been set on 2 minutes for testing purpose. And as we mentioned before, the expiration times should be loaded from the Stroke protocol.

D.3.3 First Revision Data & Process model system framework

The scenario is very similar to the scenarios above. The main difference is that this model receives not 1 or 2 messages but 3. Therefore in the model there will appear a sub-process for each kind of message like on the picture. This sub-process contains 2 other sub-processes, one with intermediate timer event. In that case the model monitors if time for incoming message is not running out. This sub-process contains tasks for the receiving message. If time runs out, the second sub-process is started. This second sub-process contains tasks for re-setting the expiration time in the Intermediate event timer (first sub-process) and the task to notify to the responsible person if the message did not come on time. The top level sub-process provides cycling over the other two sub-processes.

So there are 3 sub-process constructions in the model for 3 kinds of messages in order: LabOrder, LabCheckin and LabResult msg. The sub-process for LabCheckin msg is slightly different because A&E nurse should be notified if LabCheckin msg arrives. So there is a notification for A&E nurse in this part of the model.



“LabOrder msg” has these elements:

- ID's (Patient Id + Visit Id)
- Patient demographics
- Order ID

- Order date and time
- Ordered examination
- Collected sample ID's
- Doctor in charge of patient

“LabCheckin msg” has these elements:

- ID's (Patient Id + Visit Id)
- Patient demographics
- Order ID
- Order status
- Collected sample ID's
- Samples status
- CheckIN date and time
- CheckIN author

“LabResults msg” has these elements:

- ID's (Patient Id + Visit Id)
- Order ID
- Order status
- Tests code
- Tests result

- Test status
- Tests completion date and time
- Author of the tests

More detailed description of these messages you can find it in document “ReMINE_D7.2_NoemaLife_WP7_V1.0_Final.pdf”.

5.4.9 “ECGandAdministrationOfReperfusion” model

This part of the scenario is different. There are 3 different messages: ‘ECGExecution’, ‘Treatment’ and ‘AandEDischarge’. There is no time out for ‘ECGExecution’ message, but it must be received before one of ‘Treatment’ or ‘AandEDischarge’ message is received. If it’s not received before, ReMINE should alert to the responsible person. Then, if in 70 minutes from the start of triage it is not received, one of ‘Treatment’ or ‘AandEDischarge’ message ReMINE should alert to the responsible person. In case that one of the monitoring activity message is received, the process ends. If ‘AandEDischarge’ message doesn’t arrive in another 10 minutes after the first 70 minutes, the model also alerts to the responsible person. Then if in 20 minutes after the first 70 minutes message ‘Treatment’ is not received, the model alerts to the responsible person. To end the monitoring activity process the model needs to receive just one of ‘AandEDischarge’ or ‘Treatment’ messages.

The model uses a parallel gateway to make 2 paths. There is intermediate message component on the first path to receive the ‘ECGExecution’ message. When this message arrives, next task will set variable ‘ECG’ on true.

“ECG execution msg” has these elements:

- ID’s (Patient Id + Visit Id)
- Patient demographics
- Examination executed
- Examination date and time
- Doctor in charge of patient

“Treatment msg” has these elements:

- ID’s (Patient Id + Visit Id)
- Patient demographics
- Treatment executed
- Treatment date and time

- Doctor in charge of patient

“A&EDischarge msg” has these elements:

- ID's (Patient Id + Visit Id)
- Patient demographics
- Clinical information collected for the patient
- Time of discharging
- Doctor in charge of patient

More detailed description of these messages you can find it in document “ReMINE_D7.2_NoemaLife_WP7_V1.0_Final.pdf”.

5.4.10 “ReAssessment” model

For now there is no scenario for this model. The only thing we know is that incoming of “ReAssessment msg” causes (if the data of msg fulfills some conditions) the end of niguarda monitoring.

“ReAssessment msg” has these elements:

- ID's (Patient Id + Visit Id)
- Patient demographics
- Clinical information collected for the patient
- Clinical information related to the executed assessment
- Time of assessment
- Doctor in charge of patient

5.5. Database (RLUS) Connection

For explaining the connection of the processes with the ReMINE database we will present the connection for one scenario, for the Sacco Hospital's processes.

5.5.1 Retrieving Patient Information from RLUS

The following figure shows the procedure to retrieve the information of an incoming patient, from the database RLUS in order to use it in the processes developed under Intalio.

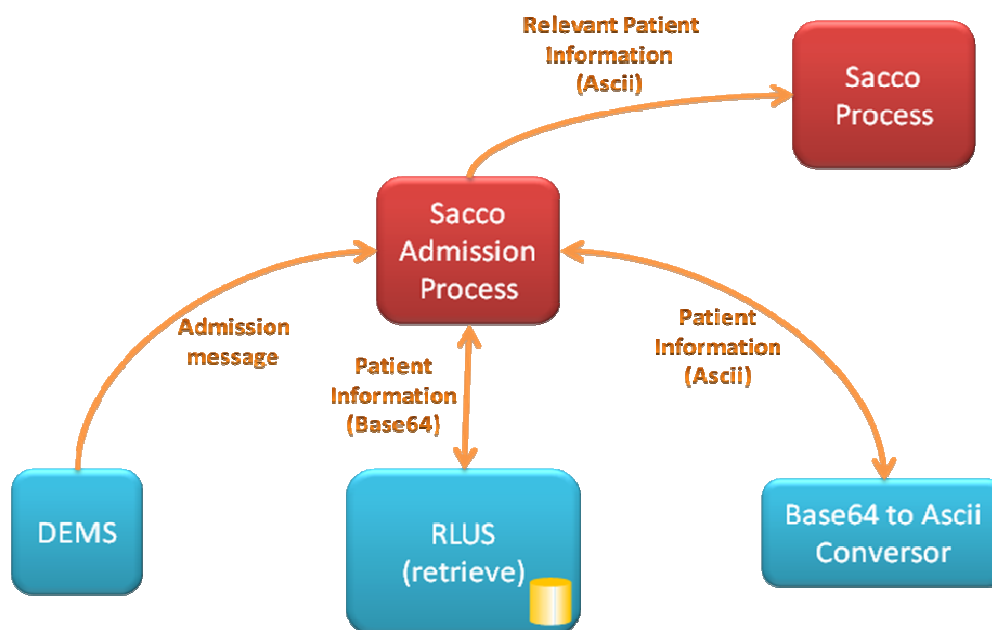


Figure 71 – Retrieving patient information from RLUS in Sacco process

The process receives messages from DEMS when an admission of a new patient takes place. This message consists on the information to locate the resource in RLUS which contains the information of the given patient. The structure of the admission message contains three elements to address the location of the resource:

1. Resource Id
2. Domain root
3. Domain extensión

With the information provided by DEMS, the process will access RLUS by means of a web service to locate the resource. This web service has been implemented by us in order to create an interface to manage RLUS functions.

RLUS returns the information of the resource in base64Binary format. The definition given by W3C says that base64Binary represents Base64-encoded arbitrary binary data. The 'value space' of base64Binary is the set of finite-length sequences of binary octets. For base64Binary data the entire binary stream is encoded using the Base64 Alphabet in [RFC 2045]. The lexical forms of base64Binary values are limited to the 65 characters of the Base64 Alphabet defined in [RFC 2045], i.e., a-z, A-Z, 0-9, the plus sign (+), the

forward slash (/) and the equal sign (=), together with the characters defined in [XML 1.0 (Second Edition)] as white space. No other characters are allowed.

As this binary information consists on xml documents, therefore we need to convert the information from base64Binary to Ascii format. This is performed through a web service. The service has been implemented by us with this aim. Once we have the resource in Ascii format consisting on a CDA or non-CDA document with xml format, we are able to extract the needed information from the document.

To extract the information we use the Mapper Palllete functions from Intalio in order to parse the xml document. Finally we can use the information of the document for our process.

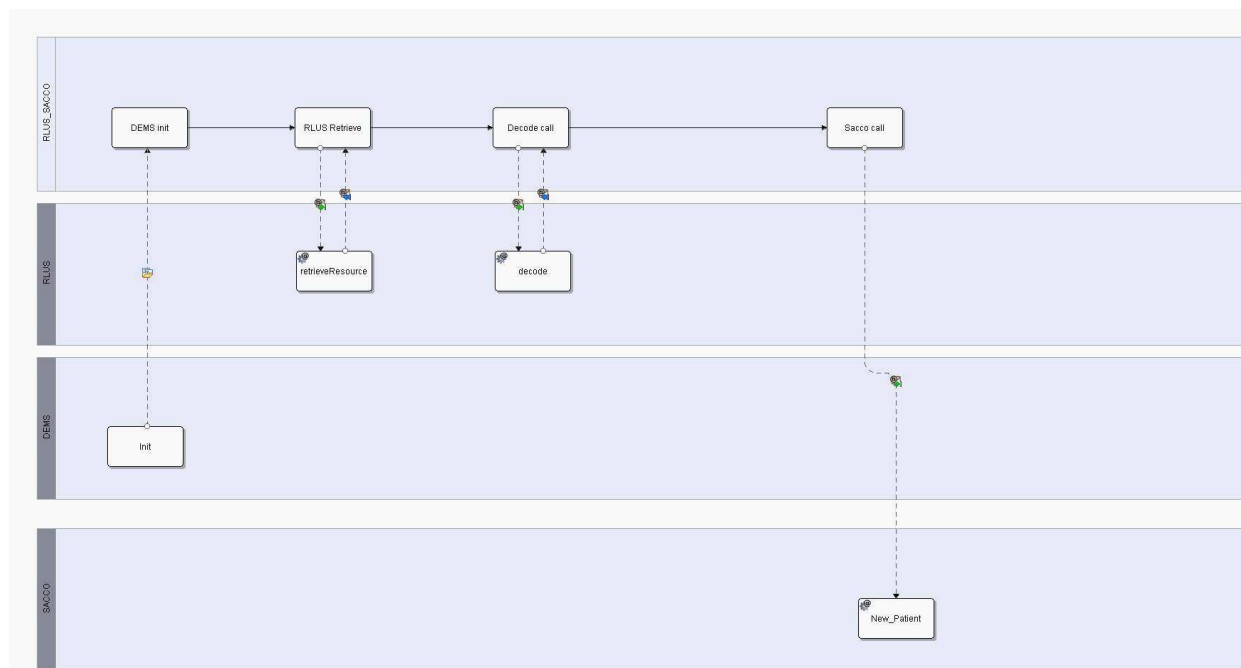


Figure 72 – Admission process for Sacco scenario.

It access to RLUS and then calls the main process of Sacco passing in the message the relevant information of the given patient

5.5.2 Updating Patient Information in RLUS

The steps to update the resources of the patients with the information submitted to the process are the next:

1. Create the new resource to store it. We have to make use of the web service to convert the xml documents to a base64Binary format in order to be able to store it in RLUS. Also the metadata has to be defined and according with the RLUS policies.
2. Check if there is a new version of the resource in order to update the last one. Through the developed web service to handle the information of RLUS we can get all the versions of the given resource. Then we get the versions from RLUS by means of the ID of our resource. We take the last entry, extracting the Id of the last version of resource (taking the node '*intenalId*').

3. Update the resource with the new one. Through the same web service used in the first step and using the operation to update resources we can update the last version of the resource. It is needed to send the new resource and the last id to relate it. In the metadata of the resource a code that states how the versioning happens has to be sent. It is sent in a 'parentAct' slot in the metadata. The codes are:

- RPLC – replace, the old resource is deleted (logically)
- APND – append

The next figure depicts all the procedure.

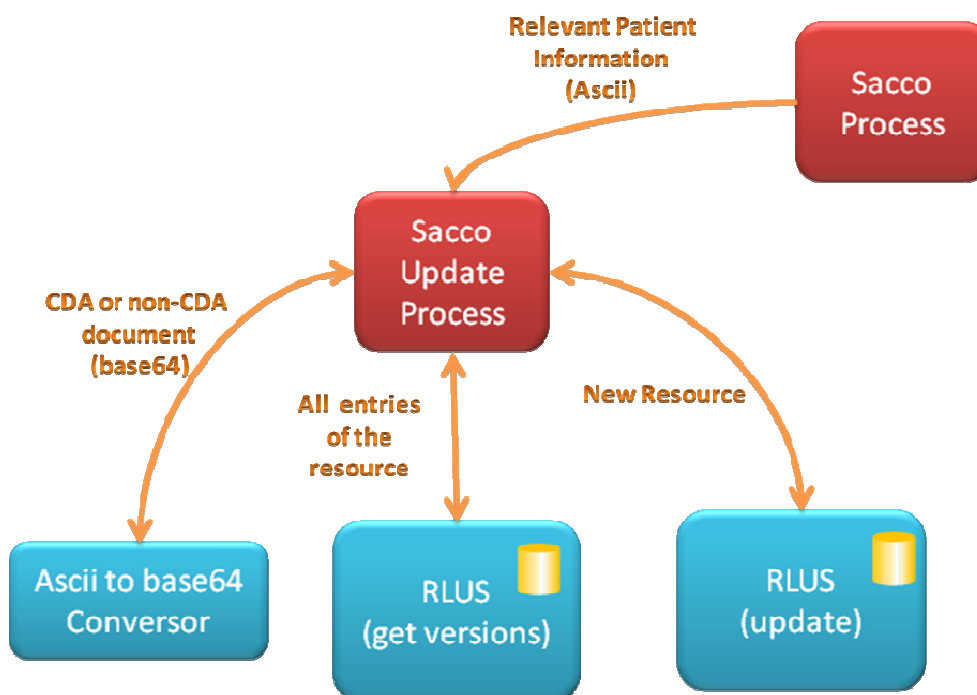


Figure 73 – Updating a resource in RLUS for Sacco scenario

5.6. Business rules Engine (BRE) integration

Business rules component is an integral part of the REMINE platform and part of the technological advantages of the project. The use of business rules within REMINE enables complex decision logic on the BPMN diagrams to be abstracted and offered in a pluggable fashion using the service oriented architecture paradigm.

For REMINE project the use of BRE provides a mechanism to evaluate several patient data conditions in a separate module and thus provide an output which can be used to control the flow inside the process.

The benefits of using BRE in REMINE are:

- A central place to store and retrieve complex rules
- Possibility to reuse rules without having to increase complexity in the processes.
- More efficient rules management (add, edit, delete)
- Data abstraction with BR implementation details hidden

D.3.3 First Revision Data & Process model system framework

For the scope of BR in REMINE and for the needs of the first and second prototype, Intalio BRE has been chosen as the technology to implement the real time rules. As seen from the following architectural diagram each process modeled in Intalio that addresses a specific clinical scenario, interfaces with an additional Intalio process that holds the business rules.

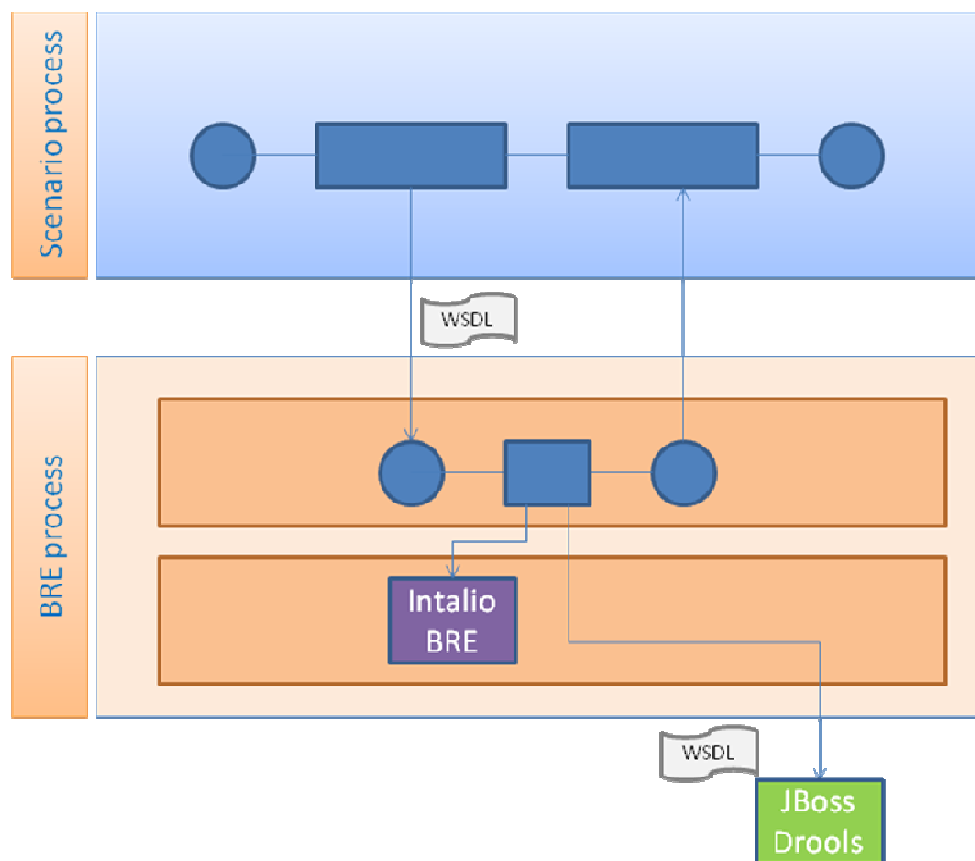


Figure 74 – Architectural diagram of BRE integration in BPEL processes

The integration is achieved by taking advantage of the supported Web Services framework from Intalio platform using an agreed interface between the involved technical partners. Every time the clinical process requires accessing a business rule the corresponding BRE process that holds the rules for each pilot will be consequently called and an output will be returned which is going to determine the course of an action. The clear separation of between control flow (clinical processes) and data logic (business rules) is handled in such a way so that changes in the implementation details on either modules does not affect the operation of the other. In addition, the BRE component can be extended in functionality or support a different technology such as JBoss Drools which is a more advanced and complete Business Rules engine.

In the following screenshot the actual Intalio BRE process is shown where it is deployed as an independent process. In this model the rules (conditions and values) that govern the Niguarda scenario are implemented using the Intalio built-in forms to create the table.

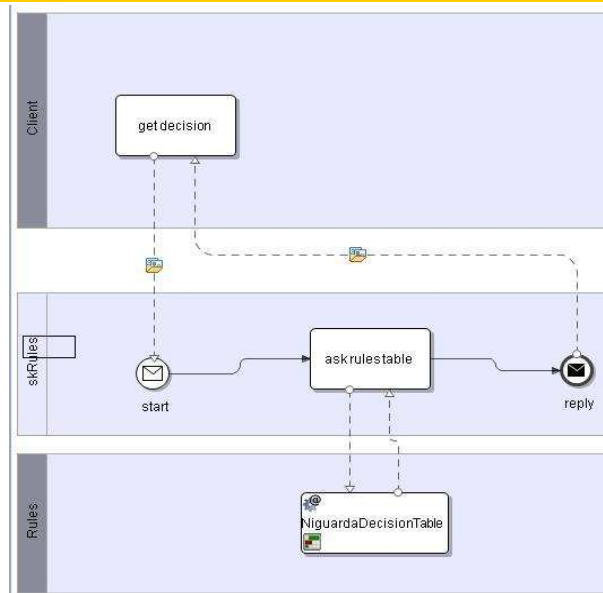


Figure 75 – BRE process

The following screenshots depicts parts of the actual NiguardaDecisionTable

ACTIONS		
Niguarda.aeTransferStrokeUnit ▼	RemAction.message = {value}: ▼	RemAction.time = {value}
*	"Neurology assessment has not been completed within 30 minutes from triage"	"PT30M"
*	"Exams have not been requested within 30 minutes from triage"	"PT30M"
*	"Cerebral CT scan has not been requested within 30 minutes from triage"	"PT30M"
*	"Test tubes have not been delivered within 15 minutes from the exam request"	"PT15M"
*	"Test tubes have been checked-in by the Lab staff"	
*	"Exam results are available"	
*	"Exam results have not been delivered within 45 minutes from the triage"	"PT45M"
*	"Contact the Laboratory for a second evaluation."	
*	"Cerebral CT image is available."	
*	"Cerebral CT scan hasn't been assessed within 60 minutes from triage."	"PT60M"
*	"Reperfusion treatment hasn't been administered within 90 minutes from triage"	"PT90M"
true	"Transfer of the patient to the Stroke Unit hasn't been done within 70 minutes from triage. Reperfusion treatment should be administered in the A&E within 20 minutes."	"PT70M"
false	"Reperfusion treatment has to be administered within 20 minutes. An eventual transfer of the patient to the Stroke Unit has to be done now."	"PT70M"

CONDITIONS						
Niguarda.priorityCode ▼	Niguarda.patientExit ▼	Niguarda.neuroAssessment ▼	Niguarda.examsComplete ▼	Niguarda.cerebralCTScanCompleted ▼	Niguarda.testTubesDeliveryToLab ▼	Niguarda.testTubesCheckIn ▼
"YELLOW STROKE"	false	false	*	*	*	*
		*	false	*	*	*
		*	*	false	*	*
		*	*	*	false	*
		true	*	*	*	true
		*	*	*	*	*
		*	*	*	*	*
		*	*	*	*	*
		*	*	*	*	*
		*	*	*	*	*
		*	*	*	*	*
		*	*	*	*	*

Figure 76 – Parts of a decision table (NiguardaDecisionTable)

6. Conclusion

6.1. Summary

Deliverable D.3.3 - First Revision Data & Process Model system framework addresses both data management and process mapping components within the 1st prototype and the 2nd of the ReMINE project. The deliverable builds up on top of the specifications given in Deliverable D3.5 - First Revision of the WP3 framework and addresses the issues that appear. The present deliverable outlines metadatabase enhancements such as extending the semantic information that TOS (Taxonomy and Ontology Service) can offer through custom-defined concept properties and as the introduction of the Master Patient Index Service that provides the patient management component for the ReMINE platform.

The process map shows all steps and the activities together with inputs and outputs. The scope of the process mapping is to provide a unifying vision of business processes, so that the organization and the individuals have a common understanding. The “Process Mapper” serves two main functions the mapping of hospital’s processes and the association of those with data and Data Events. First the processes are modeled in BPMN format, then are transformed in BPEL format for being executable and be able to trigger the appropriate web services and finally are associated to data and Data Events and are enriched with KPIs (Key Performance Indicators). This module is dependent from the ReMINE sub-module “RAPS Management System” where the execution of the processes (BPEL transformation) is implemented.

6.2. Next steps and activity plan

After receiving and evaluating the results from the three pilot sites, the third revision of the Data & Process model System framework must be realized, being due to M32. The revision must resolve the problems that arise from the 1st prototype deployment. Another deliverable is the second revision of the deliverable D.3.2 Classification and identification risk event using BPM process that is closely related to the work done for this deliverable.

Regarding the next steps required for process execution:

- 1) Association with real data and Data Events
- 2) Identification of Key Performance Indicators and integration in the processes for Reports Export. This will be helpful for the risk manager to have a view about Process performance. For every KPI the RM must have the possibility to custom the view, (pie chart, bars, etc).
- 3) Integration with the alerting system
- 4) Integration with the Business Rules Engine.

7. Glossary

A&E-Accident and Emergency

ASCII- American Standard Code for Information Interchange

API – Application Programming Interface

BPD-Business Process Diagram

BPEL-Business Process Execution Language

BPM-Business Process Management

BPMN-Business Process Modeling Notation

CDA - Clinical Document Architecture

CT-Computed tomography

CTS - Common Terminology Service

DEMS - Data Event Management System

DOM-Document Object Model

ECG-Electrocardiogram

EFM- Electronic Fetal Monitoring

EIS – (HSSP) Entity Identification Service

FHR-Fetal Heart Rate

GUI-Graphical User Interface

HIS - hospital information system

HL7 - Health Level Seven

HSSP - Healthcare Services Specification Project

KPI - Key Process Indicator

MPI - Master Patient Index

RAPS - Risks Against Patient Safety

RFID - Radio Frequency Identification

RIM - Reference Information Model

RLUS - Retrieve Locate Update Service

SQL-Structured Query Language

TOS – Taxonomy and Ontology Service

UML - Unified Modeling Language

URL- Uniform Resource Locator

WCF - Windows Communication Foundation

XML-Extensible Markup Language