ReMINE

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

FP7 Contract: 216134

– Deliverable –

D.3.8 Third Revision of the data mining and knowledge extraction module

Document Information

Document Name: ReMINE_D3.8_IW_WP3

Revision: **V0.4**

Revision Date: **08/02/2011**

Author: Adrian Podea, Maria Roditis (Info World)

Contributors: -

Security: Confidential (Consortium Only)

Document Information

Consortium, European Commission

Approvals

	Name	Beneficiary	Date	Visa
Project Coordinator	Michele CARENINI	NOEMALIFE		
Technical Manager	Fernando GUMMA	NOEMALIFE		

Document History

Revision	Date	Modification	Author
Version 0.1	26/01/2011	ToC first draft. Introduction chapters	IW
Version 0.2	03/02/2011	Data Mining chapters	IW
Version 0.3	07/02/2011	Knowledge Inference chapters	IW
Version 0.4	08/02/2011	Final review	IW

Table of Contents

1. Executive summary

Deliverable D3.8 Third revision of the data mining and knowledge extraction module describes the data mining approach used by the ReMINE platform to derive models from clinical data and also the knowledge inference module that is used for detecting risk situations.

- **Architectural overview** describes how the Data Mining component and the Knowledge Inference Systems are integrated within the ReMINE platform
- **Data mining system enhancements** describes the development of the TRFT *data mining scenario* in which data mining techniques help to identify certain useful patterns in the patient admission and hospital acquired infection control clinical data from ReMINE. Also, the chapter presents *detailed steps* for developing the data mining component in order to fulfill the scenario requests. *The results* of the data mining process are also outlined, as they will be subsequently used by the Knowledge Inference System.
- **Knowledge inference system enhancements** details the implementation of the detection processes for identifing risk situations in newly incoming clinical data, based association rule model derived through data mining. Detailed information about the *integration of the mining* models, along with the KIS architecture overview, the inference engine implementation and the KIS service contracts is provided
- **Risk Manager Interface** The implementation of the component which handles the visual delivery of the alerts to the end user is presented, including details about the service interface.

The third deliverable reflects the achievements in implementing new scenarios for the Data Mining component, integrated with the detection capabilities of the Knowledge Inference System, along with the Risk Manager Interface used to display the risk notifications. The association model obtained through data mining from the clinical data is used for detecting patient admission-related parameters that may be evidence of a situation exposing risk for hospital-acquired infection. The Risk Manager Interface was also developed and interates with the other components, as documented in the present deliverable.

2. Introduction

2.1. Deliverable vs. Project objectives

Deliverable D3.8 Third revision of the data mining and knowledge extraction module is the last in a deliverable trilogy for task T3.3 – Data mining and knowledge extraction. Task T3.3 objectives are described as following.

Sub-Task.3.3.1 Mining and incident report system – objectives:

- Collect all the data that could be useful for analysis purpose - efficient and effective data collection and representation for the purpose of further data mining analysis

- Apply data mining and path discovery techniques in order to discover relation and "characteristic" among data- enhance data mining techniques for heterogeneous data sets.

- Define an error reporting tool- reliable module responsible for presentation of the data mining results and also for sending alerts in case of high risk case discover.

Sub-Task.3.3.2 knowledge inference system – objectives:

- Develop a framework that using the information provided by the data mining module will allow the output management on the information stored inside the Metadatabase

- Develop an user interface tool on which Health operator can participate actively to the process of knowledge extraction

- Develop an infrastructure that manages the technical interface (API) for the real time risk management system and for the prost process risk management and for the post process risk management alerting system

2.2. Deliverable structure and Partners contribution

Chapters		Responsible partner
1.	Executive summary	IW
2.	Introduction	IW
3.	Architecture overview	IW
4.	Data mining system enhancements	IW
5.	Knowledge inference system enhancements	IW
6.	Conclusions	IW
7.	Glossary	IW
8.	Bibliography	IW
9.	References	IW
10.	Annexes	IW

3. Architecture overview

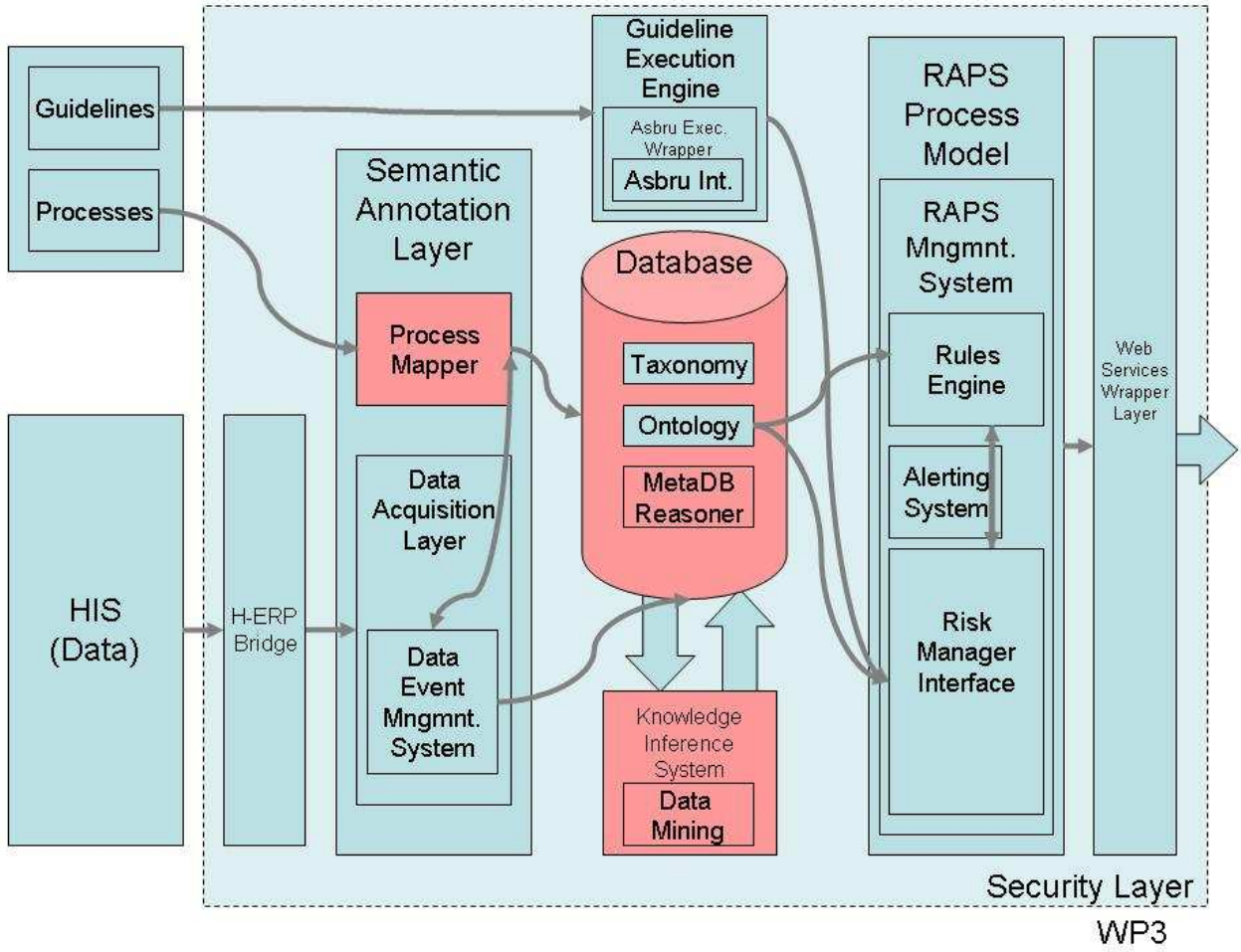

Figure 1 – The place of WP3 in ReMINE architecture

Figure 1 represents Work Package 3 components (with red) within ReMINE system. In figure 2 there is a detailed overview of the Data Mining module. This module consists of two subcomponents:

- data mining:
  - applying specific algorithms for knowledge extraction
  - rules definition (data mining results)
- knowledge inference
  - request the rules from data mining
  - apply the rules for new incoming data
  - suggest of a conclusion or decision
  - alert the risk manager

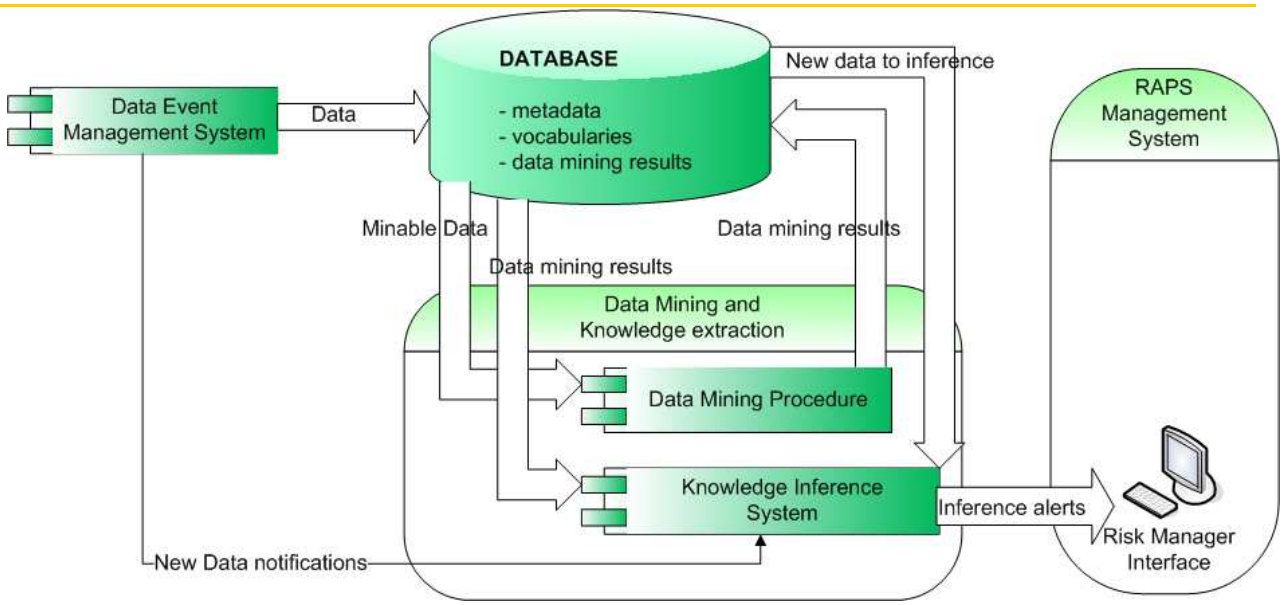

Figure 2 – Data mining and knowledge extraction module

The data mining module consists of the data mining procedure and the knowledge inference system. Data mining procedure takes all the available data in the Metadatabase (through RLUS service), applies an ETL mechanism to store the data in the database in a minable structure. For some scenarios the ETL mechanism is optional, as mining can begin on the existing metadata in the database.

The data mining results (data knowledge) consisting in the output of the data mining procedure will be stored in the database in a custom format for each data mining algorithm chosen for each scenario. The knowledge inference system will directly access the data mining results to see if any new data knowledge is available. ReMINE DEMS module will have a subscription to the Knowledge Inference System (KIS) notifying it when new data is available. KIS will process the data from the queue and send inference alerts to the Risk Manager interface.

4. Data mining system enhancements

4.1. Data mining scenarios

4.1.1. Data mining goals for TRFT pilot

Overview of business and scientific goals of the TRFT data mining scenario

The TRFT data mining pilot is focused on the medical admission process of the patients in the TRFT hospital facility and on managing the patients' risk of contracting drug-resistant bacteria, responsible with difficult-to-treat infections in humans. The phenomenon of antibiotic resistance is a high concern in current medical treatments and hospital-acquired infections are a largely recognized problem. Hospital risk managers strive to keep these categories of infections under control to mitigate risks against patient safety.

This TRFT data mining scenario within the REMINE platform aims at identifying strategic information about infection patterns in hospitals, based on thorough analysis of historical data about medical admissions and

laboratory results of infection tests. The scenario aims at providing the knowledge base for the reasoning processes performed by the KIS (Knowledge Inference System) component. The knowledge derived from data mining will enable performing the timely identification of at-risk patients and ordering appropriate preventive infection screening for them. Thus, the data mining process will be an essential auxiliary component in dealing with the spread of infections within or across hospital wards.

4.1.2. Business data mining scenarios

Patient A is admitted to the TRFT hospital for a heart attack. After receiving the initial treatment, he is allocated in a ward in the Intensive Care Unit as he still has a critical condition and needs to be continuously monitored.

In the second day from the admission, the Patient A is considered to be stable by his assigned healthcare professionals. As the Intensive Care unit occupancy rate is high, the doctors discuss the Patient A transfer to the Cardiology Ward 1 for freeing the bed in the ICU. In the Cardiology Ward 1, there is a patient infected with MRSA. The doctors would consider the risk acceptable. However, the risk manager consults the REMINE's infection prediction for the patient. REMINE's KIS system analyses the health profile of Patient A, the ward occupancy rate and the infection profile of the patients already present in the ward, against the infection patterns acquired from data mining. The KIS system concludes that Patient A has a higher risk to get infected with MRSA if transferred in the Cardiology Ward 1 and presents the justification to the risk manager through the Risk Manager Interface. On this basis, the risk manager discusses with the patient's doctors and decides to transfer the patient in a different cardiology ward.

4.1.3. Data model for Scenarios

The TRFT pilot data mining scenarios were developed based on a rigorous analysis of all available information and relationships between existing variables. Prior knowledge and in-depth analysis of the clinical processes are essential for achieving a suitable model of the problem domain.

The TRFT pilot settings and available data were synthetized in a specifically developed Application Ontology which is based and extends the more general ReMINE Taxonomy for the TRFT pilot context. The identified concepts and relationships within them were carefully analysed and are presented in in the following figures. The full OWL ontology file is delivered with the present document as **Annex 1**.

The first figure presents the hierarchical organization of the concept within the TRFT Application Ontology. The DOLCE and SHEL models are both represented as in the initial ReMINE ontology and augmented with particular concepts for the hospital admission and infection control processes.

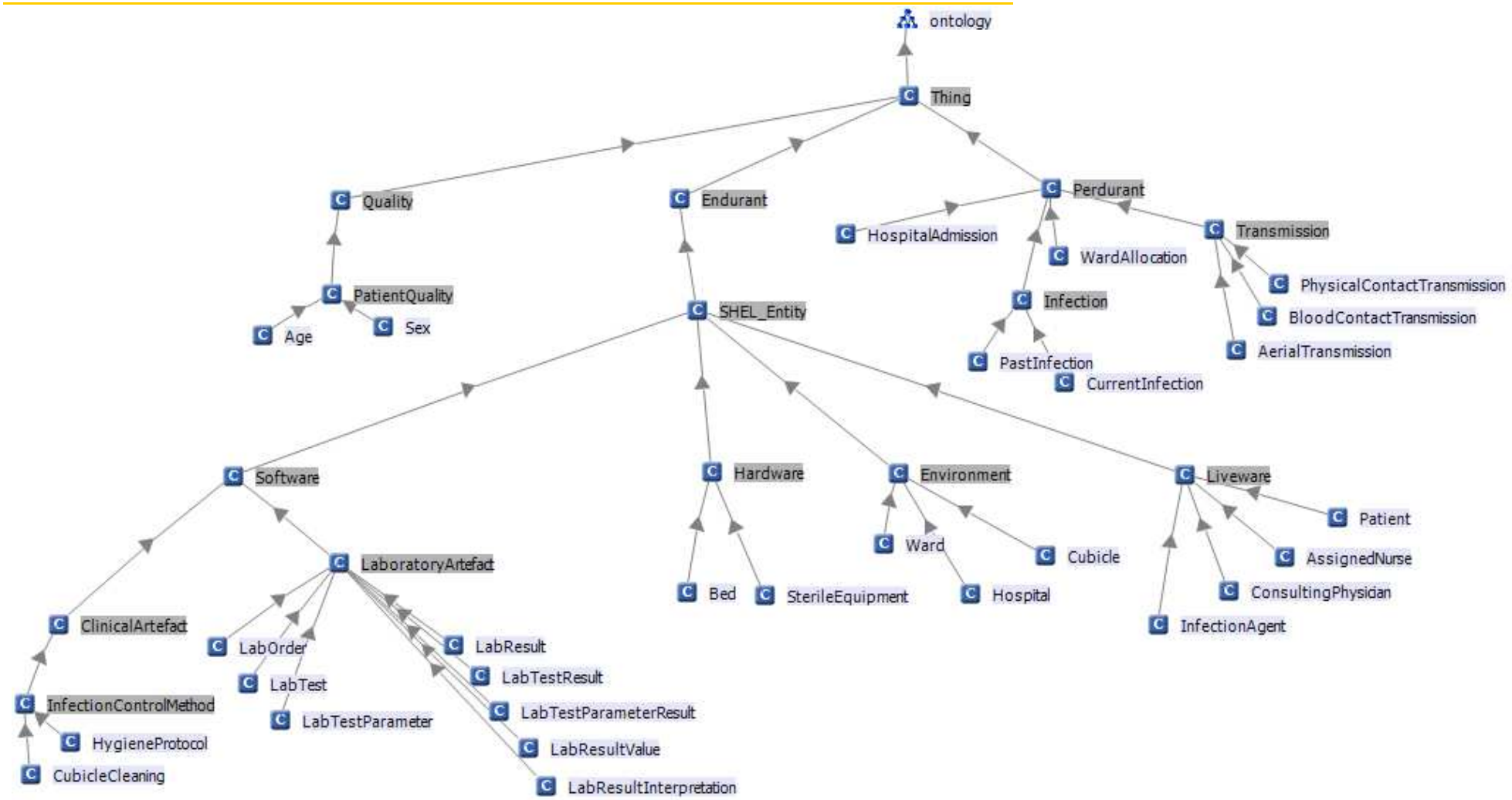

Figure 3. Hierarchical organization of concepts in TRFT Application Ontology

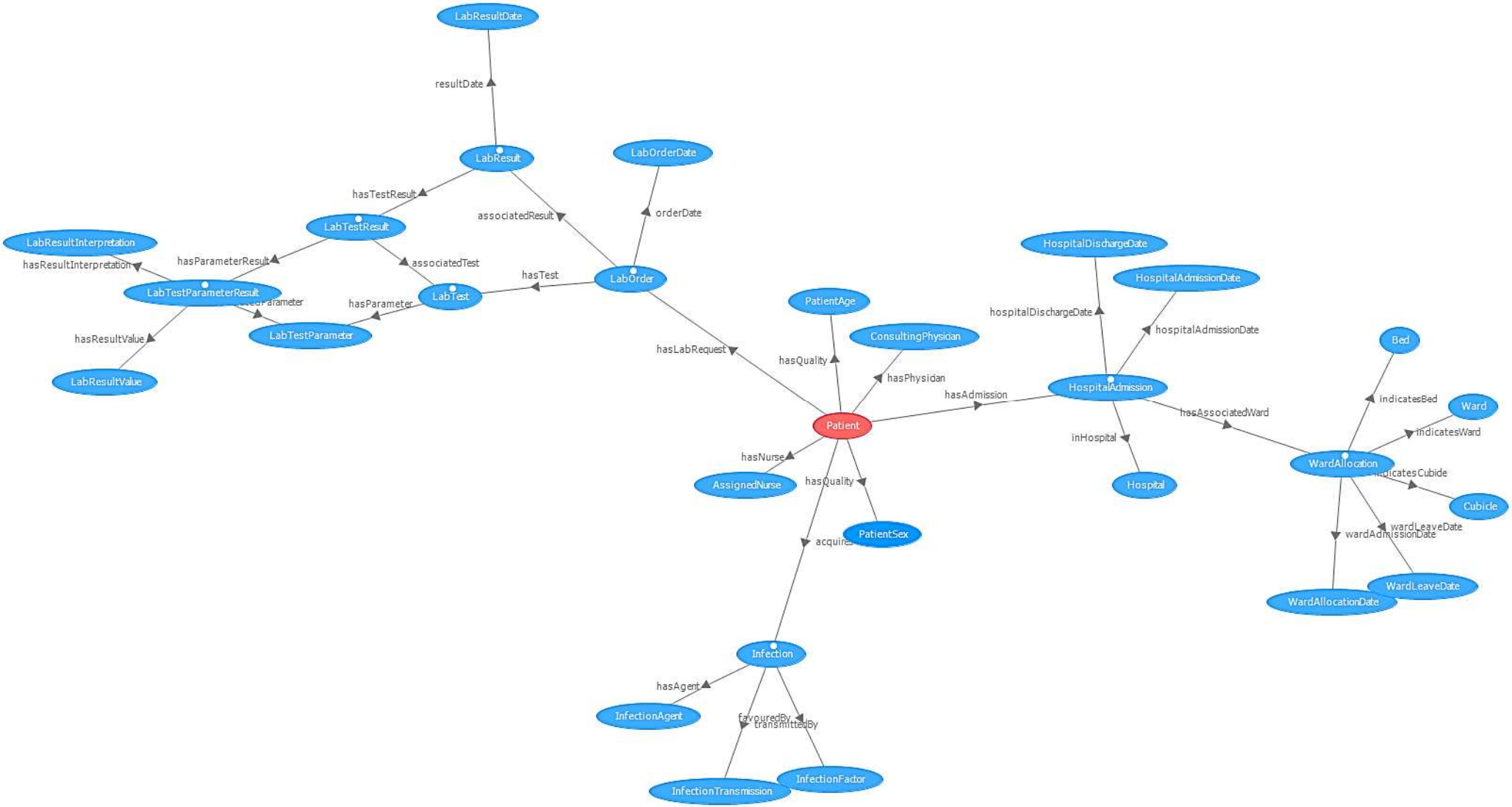

Figure 4. Semantic relationships between concepts in the TRFT Application Ontology

Figure 4 presents the relationships between the concepts in the ontology, expressing how variable within obtained historical data are related to each other. Based on modelling the prior knowledge with an application ontology, the development of the data mining scenarios is performed. The scenarios are presented in the following.

The available clinical data which is used for the data mining and knowledge inference process is described in the following. Data arrives on the ReMINE platform from the Patient Administration System (PAS) and Laboratory Information System (LIS) systems from the TRFT pilot HIS.

A short overview of the data contained within the types of transferred messages from PAS and LIS to ReMINE is given in the table below.

EVENT	AVAILABLE	SENDER	MESSAGE	HL7
Patient Admission [TRFT-01]	Yes	PAS	ADT^A01	ADT^A01
Patient Update [TRFT-02]	Yes	PAS	ADT^A08	ADT^A08
~~Lab Order [TRFT-03]~~	~~Not yet~~	~~PAS~~	~~ORM^O01~~	~~OML~~
Lab Result [TRFT-04]	Yes	LIS	ORU^R01	ORU^R01
Patient Transfer [TRFT-05]	Yes	PAS	ADT^A02	ADT^A02
Patient Discharge [TRFT-06]	Yes	PAS	ADT^A03	ADT^A03
~~Task Executed [TRFT-07]~~	~~NO~~			~~SDN/SCN~~

TRFT added by 21st of June 2010 another types of messages related to the management of patients, as in the next table.

EVENT	AVAILABLE	SENDER	MESSAGE	HL7
Patient Registration	Yes	PAS	ADT^A28	ADT^A28
Patient Details Update	Yes	PAS	ADT^A31	ADT^A31
Patient Merge	Yes	PAS	ADT^A34	ADT^A34
Change Patient ID	Yes	PAS	ADT^A47	ADT^A47

The data included in the messages is presented by each message type in the next table.

DATA	CARDINALITY	Use
Patient Admission [TRFT-01]		
ID's (Patient Id + Visit Id)	1-*	required
Patient demographics (Date of Birth, Sex)		required
Patient address	1	required
MRSA/other infection notes	1	required
Admitting ward.	1	
Room and bed assignment	1	required

Patient Update [TRFT-02]		
ID's (Patient Id + Visit Id)	1-*	required
Patient demographics (Date of Birth, Sex)		required
Patient address	1	required
MRSA/other infection notes	1	required
Admitting ward.	1	
Room and bed assignment	1	required
Date and Time of admission	1	required
Lab Result [TRFT-04]		
ID's (Patient Id + Visit Id)	1-*	required
Patient demographics		required
Order ID	1	required
Order date and time	1	required
Ordered Tests e.g MRSA	1	required
Ordering Ward	1	required
Ordering Person Name	1	required
Order status	1	required
Test performed	1-*	required
Test results	1-*	required
Test execution date and time	1-*	required
Patient Transfer [TRFT-05]		
ID's (Patient Id + Visit Id)	1-*	required
Patient demographics		required
Patient address	1	required
MRSA/other infection notes	1	required
Old ward.	1	required
New Ward	1	required
Room and bed assignment	1	
Date and Time of transfer	1	required
Patient Discharge [TRFT-06]		
ID's (Patient Id + Visit Id)	1-*	required
Patient demographics (Date of Birth, Sex)		required
Patient address	1	required
MRSA/other infection notes	1	required
Discharging ward.	1	
Room and bed assignment	1	required
Date and Time of discharge	1	required

Data Mining Scenario Details

The TRTF mining scenario is targeted at predicting the infection probability of a certain patient category, based on infection information about the neighbour patients in the same ward. This approach is motivated by the fact that infections can be spread by nurses or doctors when passing from a patient to another, if they neglect hygiene standards such as washing hands and/or use sterile gloves etc.

Thus, the data mining scenario exploits the information about the previous and current infections of the patients in the same ward. The reason for which patients' previous infections with drug-resistant bacteria are also considered is the fact that treating an infection and obtaining a negative result in testing is a probable but not a definitive proof for successfully treating the infection, as bacteria may change the infection site and be overlooked when testing.

The data mining scenario aims at modelling the probability of a patient becoming infected, based on the infected persons in the current and previous wards he is/was located in. A weighting of the exposure risk is performed in concordance with the length of time spent in each ward.

The data mining approach considers information such as:

- Patient demographic category (age and sex information). Patients within certain age group, for example, may be predisposed to infection spread, such as in case of young children or elderly, who may have lower immunity.
- Patient's previous infection notes, as they may be a sign of a weaken organism or may reveal a recurring infection
- The infection profiles of the patient's neighbours, based on the collected ward allocation information.
- Moreover, the infection process may be influenced by the length of stay of the patient in the hospital. A longer stay may increase the risk of acquiring the infection from his ward / hospital neighbour patients.

4.2. Data mining implementation

The data mining process includes a sequence of steps for the TRFT mining scenario as following:

- Sourcing and saving data into the RLUS Metadatabase
- Data acquisition
- Data anonymization
- Data pre-processing
- Specifying the data sources
- Building the Association Mining Model
- Assessing algorithm parameters
- The data mining process
- Evaluation of results & retuning parameters

The overview of the mining process, with specific aspects of the TRFT scenario, is presented below:

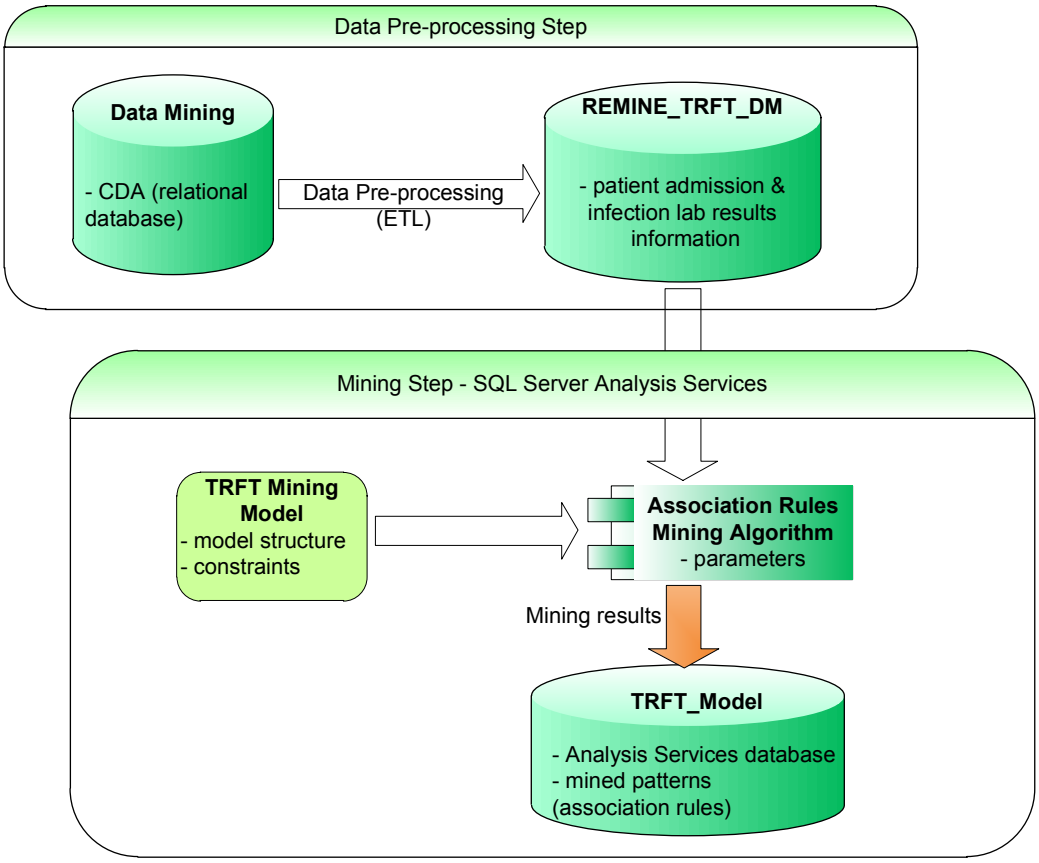

Figure 5. Overview of TRFT data mining process

4.2.1. Source data for mining

The HIS data is sourced on the ReMINE Platform by using the H-ERP bridge, passed through the Data Acquisition Layer and stored into the RLUS database. However, the data from HIS must endure some transformations to become conformant to the CDA standards enforced by the RLUS repository. Every data mining scenario must define its own data requirements with respect to the HIS data contract and the CDA document specifications.

The CDA specifications for the patient administration and laboratory result sections that the RLUS repository uses are presented further. The CDA header implementation employed in the clinical documents for TRFT Pilot was already annexed to an earlier deliverable, D3.5, as mentioned in the current document's Reference 1.

Table 1 – **Patient Hospital Admission Section** (CDA specifications used in the RLUS repository)

Cardinality: 0..1

Description

This section describes the attributes of the patient admission in the hospital facility

Elements and attributes

Element	Attributes	Data Type	Cardinality	Description
Component.section.code	code codeSystem codeSystemName displayName	CE	1..1	The code for the patient hospital admission section **ValueSet:** LOINC **Fixed value**: 29554-3 **Example:** <code code="29554-3" codeSystem="2.16.840.1.113883.6.1" codeSystemName="LOINC" displayName="Procedure" />
component.section.title		ST	0..1	Section title **Fixed value:** Patient Admission
Component.section.text		text	0..1	This is where human-readable information is inserted
Component.section.entry.procedure	classCode moodCode			Admission details. Attributes: classCode="PROC" moodCode="EVN"
component.section.entry. procedure.code	code codeSystem codeSystemName displayName	CE	1..1	Code for the hospital admission **ValueSet:** SNOMED-CT **Fixed code:** 32485007 **Example:** <code code="32485007" codeSystem="2.16.840.1.113883.6.96" codeSystemName="SNOMED-CT" displayName="Hospital admission" />

component .section.entry.procedure.statusCode	code="completed"	CNE	0..1	Inserted for the completed hospital admission
component .section.entry.procedure.effectiveTime	value	TS	0..1	Date of the patient admission

Table 2 – **Patient Admission to Ward Section** (CDA specifications used in the RLUS repository)

Cardinality: 0..1

Description

This section describes the attributes of the patient admission to the ward

Elements and attributes

Element	Attributes	Data Type	Cardinality	Description
Component.section.code	code codeSystem codeSystemName displayName	CE	1..1	The code that identifies the patient ward admission section **ValueSet:** SNOMED-CT **Fixed value**: 225746001 **Example:** <code code="225746001" codeSystem="2.16.840.1.113883.6.96" codeSystemName="SNOMED CT" displayName="Ward">
Component.section.id	Root Extension	II	0..1	The admission ward is indicated here. **Root**=2.16.840.1.113883.3.270 **Extension**= ward identifier **Example:** <id extension="CARD" root="2.16.840.1.113883.3.270" />

Component.section.code.qualifier	Name Value	CR	0..1	Indicates whether this is a past or current ward admission
Component .section.code.qualifier.name	code codeSystem codeSystemName displayName	CE	0..1	The name indicates the semantics of the qualifier (current or past admission) **ValueSet:** SNOMED-CT **Fixed Value:** 410511007 Example: <name code="410511007" codeSystem="2.16.840.1.113883.6.96" codeSystemName="SNOMED CT" displayName="Current or past" />
Component.section.entry. code.qualifier.value	code codeSystem codeSystemName displayName	CE	0..1	The value indicates a past/current concept associated with admission. **ValueSet:** SNOMED-CT **Fixed Value:** 410513005 Example: <value code="410513005" codeSystem="2.16.840.1.113883.6.96" codeSystemName="SNOMED CT" displayName="Past" />
component.section.title		ST	0..1	Ward Admission Section title
Component.section.text		text	0..1	This is where human-readable information is inserted
Component.section.entry.procedure	classCode moodCode			Ward admission details. Attributes: classCode="PROC" moodCode="EVN"
component.section.entry. procedure.code	code codeSystem codeSystemName displayName	CE	1..1	Code for the ward admission **ValueSet:** SNOMED-CT **Fixed code:** 305342007 **Example:** <code code=" 305342007 " codeSystem="2.16.840.1.113883.6.96 codeSystemName="SNOMED-CT" displayName=" Admission to ward " />
component .section.entry.procedure.statusCode	code="completed"	CNE	0..1	Inserted for the completed ward admission

component .section.entry.procedure.effectiveTime	value	TS	0..1	Date of the ward admission

Table 3 – **Patient Discharge Section** (CDA specifications used in the RLUS repository)

Cardinality: 0..1

Description

This section describes the attributes of the patient discharge section

Elements and attributes

Element	Attributes	Data Type	Cardinality	Description
Component.section.code	code codeSystem codeSystemName displayName	CE	1..1	The code that identifies the patient discharge section **ValueSet:** LOINC **Fixed value**: 29554-3 **Example:** <code code="29554-3" codeSystem="2.16.840.1.113883.6.1" codeSystemName="LOINC" displayName="Procedure" />
component.section.title		ST	0..1	Patient discharge section title
Component.section.text		text	0..1	This is where human-readable information is inserted
Component.section.entry.procedure	classCode moodCode			Discharge details. Attributes: classCode="PROC" moodCode="EVN"

component.section.entry. procedure.code	code codeSystem codeSystemName displayName	CE	1..1	Code for the discharge procedure **ValueSet:** SNOMED-CT **Fixed code:** 58000006 **Example:** <code code=" 58000006" codeSystem="2.16.840.1.113883.6.96 codeSystemName="SNOMED-CT" displayName=" Patient discharge" />
component.section.entry. procedure.code.qualifier.name	code codeSystem codeSystemName displayName	CE	0..1	The name indicates the semantics of the qualifier (type of discharge) **ValueSet:** SNOMED-CT **Fixed Value:** 107726003 Example: <name code=" 107726003 " codeSystem="2.16.840.1.113883.6.96" codeSystemName="SNOMED CT" displayName=" Patient disposition" />
component.section.entry. procedure.code.qualifier.value	code codeSystem codeSystemName displayName	CE	0..1	The value indicates the type of discharge **ValueSet:** NiguardaDischargeDisposition **Fixed Value:** 1 Example: <value code="1" codeSystem="100.100.100.50" codeSystemName="NiguardaDischargeDisposition" />
component .section.entry.procedure.statusCode	code="completed"	CNE	0..1	Inserted for the completed discharge
component .section.entry.procedure.effectiveTime	value	TS	0..1	Date of the discharge

Table 4 – **Laboratory Results Section** (CDA specifications used in the RLUS repository)

Cardinality: 0..1

Description:

This section describes the attributes of the laboratory results section

Elements and attributes

Element	Attributes	Data Type	Cardinality	Description
Component.section.code	code codeSystem codeSystemName displayName	CE	1..1	The code that identifies the laboratory results section **ValueSet:** LOINC **Fixed value**: 34075-2 **Example:** <code code="34075-2" codeSystem="2.16.840.1.113883.6.1" codeSystemName="LOINC" displayName="Laboratory Results" />
component.section.title		ST	0..1	Laboratory results section title
Component.section.text		text	0..1	This is where human-readable information is inserted
Component.section.entry.procedure	classCode moodCode			Associated laboratory order details Attributes: classCode="PROC" moodCode="EVN"
Component.section.entry.procedure.id	Root Extension	II	0..1	The order number is indicated here. **Root**= 2.16.840.1.113883.3.270 **Extension**= Order number **Example:** <id extension="0010U104300" root="2.16.840.1.113883.3.270" />
component.section.entry.procedure.statusCode	code="completed"	CNE	0..1	Inserted for the completed order
component.section.entry.procedure.effectiveTime	value	TS	0..1	Date and time of order

component.section.component.section.code	code codeSystem codeSystemName displayName	CE	1..1	Code for the requested laboratory test **Example:** <code code="301786007" codeSystem="2.16.840.1.113883.6.96" codeSystemName="SNOMED-CT" displayName="Multi-resistant staphylococcus aureus screening">
component.section.component.section.code.qualifier.name	code codeSystem codeSystemName displayName	CE	0..1	Qualifier indicates whether the patient lives in a residential home. **ValueSet:** SNOMED-CT **Fixed Value:** 394923006 **Example**: <name code="394923006" codeSystem="2.16.840.1.113883.6.96" codeSystemName="SNOMED-CT" displayName="Lives in a residential home" />
component.section.component.section.code.qualifier.value	code codeSystem codeSystemName displayName	CE	0..1	Indicates a Boolean value: True/False **Example**: <value code="64100000" codeSystem="2.16.840.1.113883.6.96" codeSystemName="SNOMED-CT" displayName="False" />
component.section.component.section.component.section.code	displayName code codeSystem codeSystemName	CE	1..1	Laboratory test result **Example**: <code code="34075-2" codeSystem="2.16.840.1.113883.6.96" codeSystemName="SNOMED-CT" displayName="Methicillin resistant Staphylococcus aureus infection">
component.section.component.section.component.section.code.qualifier.name	code codeSystem codeSystemName displayName	CE	0..1	Qualifier indicates whether the laboratory test is positive or negative **ValueSet:** SNOMED-CT **Fixed Value:** 394923006 **Example**: <name code="394923006" codeSystem="2.16.840.1.113883.6.96" codeSystemName="SNOMED-CT" displayName="Lives in a residential home" />
component.section.component.section.component.section.code.qualifier.value	code codeSystem codeSystemName displayName	CE	0..1	Indicates a Boolean value: True/False **Example**: <value code="260385009" codeSystem="2.16.840.1.113883.6.96" codeSystemName="SNOMED-CT" displayName="Negative" />

component.section.component.section.component.section.entry.observation.id	Root Extension	II	0..1	Indicates the code of the requested test **Root =** 2.16.840.1.113883.3.270 **Extension =** Request code **Example:** <id extension="MR" root="2.16.840.1.113883.3.270" />
component.section.component.section.component.section.entry.observation.code	displayName code codeSystem codeSystemName	CE	1..1	Indicates the class of the requested test **Example**: <code code="19851009" codeSystem="2.16.840.1.113883.6.96" codeSystemName="SNOMED-CT" displayName="Microbiology test">
component.section.component.section.entry.observation.code.originalText		ST	0..1	Information about the request **Example**: <originalText> <reference value="MRSA" /> </originalText>
component.section.component.section.entry.observation.statusCode	code="completed"	CNE	0..1	Inserted for the completed laboratory result
component.section.component.section.entry.observation.effectiveTime.low, high	value	IVL_TS	0..1	Date of observation start and end respectively
component.section.component.section.entry.observation.targetSiteCode	code codeSystem codeSystemName displayName	CE	0..1	Specimen Source Description **Example:** <targetSiteCode code="39937001" codeSystem="2.16.840.1.113883.6.96" codeSystemName="SNOMED-CT" displayName="Skin" />
component.section.component.section.entry.observation.author.time	value	TS	0..1	Date and time of order

component.section.component.section.entry.observation.author.assignedAuthor.id	Root Extension	II	0..1	The identifier of the clinician requesting the laboratory test **Root**= 2.16.840.1.113883.3.270 **Extension**= Requesting Clinician Identifier **Example:** <id extension="AAL" root="2.16.840.1.113883.3.270" />
component.section.component.section.entry.observation.author.assignedAuthor.assignedPerson. name	family given	ON		The name of the clinician requesting the laboratory test.

Example 1. CDA containing ADT (Admission, Discharge and Transfer) information for a patient, including the header and three sections: one Patient Hospital Admission, Admission to Ward and Patient Discharge sections.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<ClinicalDocument xmlns="urn:hl7-org:v3"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
  <typeId extension="POCD_HD000040" root="2.16.840.1.113883.1.3" />
  <templateId root="2.16.840.1.113883.3.27.1776" />
  <id extension="1234567-12345" root="2.16.840.1.113883.3.270.1.1" />
  <code code="51855-5" codeSystem="2.16.840.1.113883.6.1"
    codeSystemName="LOINC" displayName="Clinical note" />
  <effectiveTime value="20100712161700" />
  <confidentialityCode code="N" codeSystem="2.16.840.1.113883.5.25"
    codeSystemName="Confidentiality" displayName="Normal" />
  <setId extension="AB1234" root="2.16.840.1.113883.3.270.1.2" />
  <versionNumber value="1" />
  <recordTarget>
    <patientRole>
      <id extension="AA2533" root="2.16.840.1.113883.3.270.1.3" />
      <addr>
        <streetName>STREET NAME</streetName>
        <city>CITY NAME</city>
      </addr>
      <patient>
        <name>
          <given>GIVEN NAME</given>
          <family>FAMILY NAME</family>
        </name>
        <administrativeGenderCode code="F"
          codeSystem="2.16.840.1.113883.5.1" codeSystemName="AdministrativeGender"
          displayName="Female" />
        <birthTime value="19400113" />
      </patient>
    </patientRole>
  </recordTarget>
  <author>
    <time value="20100712161700" />
    <assignedAuthor>
      <id extension="TC-PAS" root="2.16.840.1.113883.3.270" />
      <representedOrganization>
        <id root="2.16.840.1.113883.3.270" />
      </representedOrganization>
    </assignedAuthor>
  </author>
  <custodian>
    <assignedCustodian>
      <representedCustodianOrganization>
        <id root="2.16.840.1.113883.3.270" />
        <name>TC</name>
      </representedCustodianOrganization>
    </assignedCustodian>
  </custodian>
  <componentOf>
    <encompassingEncounter>
      <code code="338709012" codeSystem="2.16.840.1.113883.6.96"
        codeSystemName="SNOMED-CT" displayName="Triage" />
      <effectiveTime value="20100709" />
```

```
    </encompassingEncounter>
  </componentOf>
  <component>
    <structuredBody>
      <component>
        <section>
          <code code="29554-3" codeSystem="2.16.840.1.113883.6.1"
            codeSystemName="LOINC" displayName="Procedure" />
          <title>Patient Admission</title>
          <entry>
            <procedure classCode="PROC" moodCode="EVN">
              <code code="32485007" codeSystem="2.16.840.1.113883.6.96"
                codeSystemName="SNOMED-CT" displayName="Hospital admission" />
              <statusCode code="completed" />
              <effectiveTime value="20100709" />
            </procedure>
          </entry>
        </section>
      </component>
      <component>
        <section>
          <id extension="CARD" root="2.16.840.1.113883.3.270" />
          <code code="225746001" codeSystem="2.16.840.1.113883.6.96"
            codeSystemName="SNOMED CT" displayName="Ward">
            <qualifier>
              <name code="410511007" codeSystem="2.16.840.1.113883.6.96"
                codeSystemName="SNOMED CT" displayName="Current or past" />
              <value code="410513005" codeSystem="2.16.840.1.113883.6.96"
                codeSystemName="SNOMED CT" displayName="Past" />
            </qualifier>
          </code>
          <title>Admission to ward</title>
          <entry>
            <procedure classCode="PROC" moodCode="EVN">
              <code code="305342007" codeSystem="2.16.840.1.113883.6.96"
                codeSystemName="SNOMED-CT" displayName="Admission to ward" />
              <statusCode code="completed" />
              <effectiveTime value="20100709" />
            </procedure>
          </entry>
        </section>
      </component>
      <component>
        <section>
          <code code="29554-3" codeSystem="2.16.840.1.113883.6.1"
            codeSystemName="LOINC" displayName="Procedure" />
          <title>Patient discharge</title>
          <entry>
            <procedure classCode="PROC" moodCode="EVN">
              <code code="58000006" codeSystem="2.16.840.1.113883.6.69"
                codeSystemName="SNOMED-CT" displayName="Patient discharge">
                <qualifier>
                  <name code="107726003" codeSystem="2.16.840.1.113883.6.69"
                    codeSystemName="SNOMED-CT" displayName="Patient disposition" />
                  <value code="1" codeSystem="100.100.100.50"
                    codeSystemName="NiguardaDischargeDisposition" />
                </qualifier>
              </code>
              <statusCode code="completed" />
              <effectiveTime value="20100709" />
            </procedure>
          </entry>
```

```
      </section>
    </component>
  </structuredBody>
 </component>
</ClinicalDocument>
```

Example 2. Laboratory result CDA section containing a negative result for MRSA infection:

```
<component>
  <section>
    <code code="34075-2" codeSystem="2.16.840.1.113883.6.1"
      codeSystemName="LOINC" displayName="Laboratory Results" />
    <title>Microbiology tests</title>
    <entry>
      <procedure classCode="PROC" moodCode="RQO">
        <id extension="0010U104300" root="2.16.840.1.113883.3.270" />
        <statusCode code="completed" />
        <effectiveTime value="201006291051" />
      </procedure>
    </entry>
    <component>
      <section>
        <code code="301786007" codeSystem="2.16.840.1.113883.6.96"
          codeSystemName="SNOMED-CT" displayName="Multi-resistant staphylococcus aureus
screening">
          <qualifier>
            <name code="394923006" codeSystem="2.16.840.1.113883.6.96"
              codeSystemName="SNOMED-CT" displayName="Lives in a residential home" />
            <value code="64100000" codeSystem="2.16.840.1.113883.6.96"
              codeSystemName="SNOMED-CT" displayName="False" />
          </qualifier>
        </code>
        <component>
          <section>
            <code code="34075-2" codeSystem="2.16.840.1.113883.6.96"
              codeSystemName="SNOMED-CT"
              displayName="Methicillin resistant Staphylococcus aureus infection">
              <qualifier>
                <name code="281296001" codeSystem="2.16.840.1.113883.6.96"
                  codeSystemName="SNOMED-CT" displayName="Result Comments" />
                <value code="260385009" codeSystem="2.16.840.1.113883.6.96"
                  codeSystemName="SNOMED-CT" displayName="Negative" />
              </qualifier>
            </code>
            <entry>
              <observation classCode="COND" moodCode="EVN">
                <id extension="MR" root="2.16.840.1.113883.3.270" />
                <code code="19851009" codeSystem="2.16.840.1.113883.6.96"
                  codeSystemName="SNOMED-CT" displayName="Microbiology test">
                  <originalText>
                    <reference value="MRSA" />
                  </originalText>
                </code>
                <statusCode code="completed" />
                <effectiveTime>
                  <low value="201006281532" />
                  <high value="201006280000" />
                </effectiveTime>
```

```
            <targetSiteCode code="39937001" codeSystem="2.16.840.1.113883.6.96"
              codeSystemName="SNOMED-CT" displayName="Skin" />
            <author>
              <time value="201006291051" />
              <assignedAuthor>
                <id extension="ABCD" root="2.16.840.1.113883.3.270" />
                <assignedPerson>
                   <name>
                          <given>Healthcare professional name</given>
                   </name>
                </assignedPerson>
                <representedOrganization>
                   <id root="2.16.840.1.113883.3.270" />
                </representedOrganization>
              </assignedAuthor>
            </author>
          </observation>
        </entry>
      </section>
    </component>
  </section>
 </component>
</section>
</component>
```

4.2.2. Data acquisition

The data from the RLUS Repository must be extracted, processed and stored into a general Data Mining database, which is done periodically by a specifically designed process, CDAService. Roughly, CDAService extracts clinical data from RLUS and loads it into a relational database – process presented into **Figure 6**. The resulting data can be used either for direct mining, or for further pre-processing.

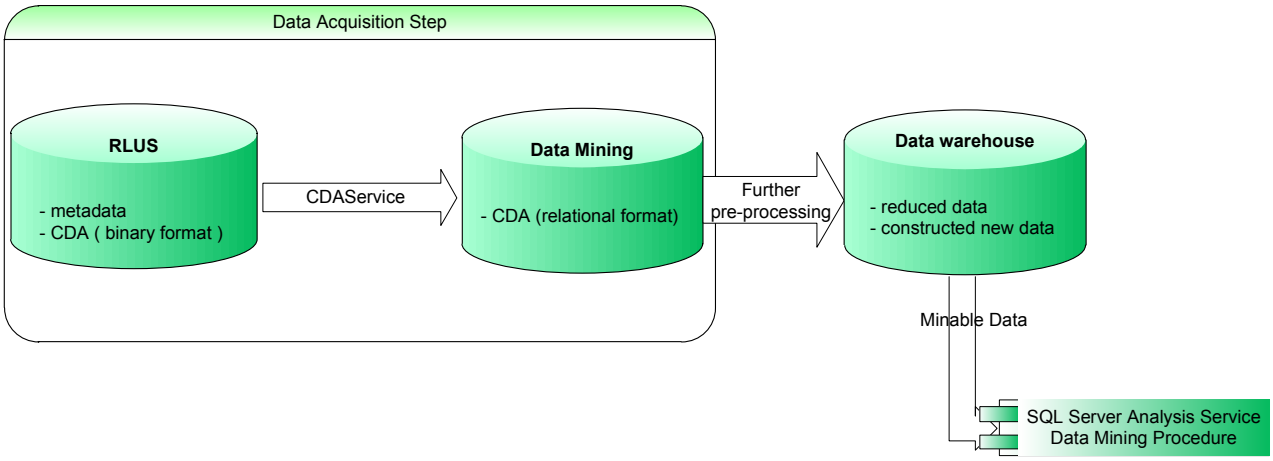

Figure 6 – Data Acquisition Step in the TRFT Data Mining Module

The CDA Service performs the data transfer between RLUS and the Data Mining database by using two steps, database generation and data persistence.

In the database generation step, a tool called CDADbGenerator has be run prior to running CDAService. The CDADbGenerator tool generates the complete structure for a SQL Server database, and also the source files for the object-relational translation process. The generated database will be the target database where the CDAService will store the transformed CDA data. The translator itself is used in the processing done by CDAService on the documents extracted from RLUS.

The source files generated for the translator are an XSD schema and a set of C# source code files. The input for generation is the data type structure of a Clinical Document as it is standardized by HL7 specifications.

In the SQL generation part, the data type is analyzed and recursively traversed. For each different HL7 CDA data type, an SQL table is created. A HL7 CDA data type can be regarded as an XML tag – to simplify explaining the generation process. As a CDA tag may also contain children tags, the children tags are transformed recursively into SQL tables. The relationship between parent and children table enforces the CDA cardinality restrictions. For example, if a parent tag contains exactly one child, the parent table receives a foreign key to the child table. If the parent tag contains a collection of the children tags of same type, the child table is the one that receives the foreign key to the parent table. Foreign keys also enable the bidirectional traversal of tables, from parents to children and backwards. Moreover, the database contains a set of stored procedures that ease various operations with the data.

An XSD schema is also generated to encode the CDA data type structure. The schema will serve for the run-time persistence process. More specifically, it is used for instantiating a System.Data.DataSet instance which will be filled with the data read from the CDA object. The DataSet will be finally saved into the database.

Furthermore, a translation component is needed for realizing the recursive traversing of the source CDA object and filling the DataSet with data. It is the reason for which the set of C# source code files are generated. They are compiled into the translator component that traverses the data object, extracts the necessary data, aggregates it and fills the DataSet object.

The data persistence step consists in converting the clinical document objects into relational data. CDAService is configured to target a database which was previously generated with the CDADbGenerator tool, as stated in the database generation step.

CDAService incrementally fills the target database with new data from the RLUS. Periodically, CDAService queries the RLUS metadatabase to identify the newly inserted / modified clinical documents. It retrieves these documents from RLUS in binary form, and executes the data persistence mechanism, converting and loading them as relational data into the target database.

As stated before, the persistence mechanism uses a complex translator and the System.Data.DataSet .net data type that encodes the CDA datatype – all of which are generated into the prior database generation step. The translator parses and transforms the data, returning an instance of DataSet containing object data. The DataSet instance is subsequently saved into the database thus creating an identical relational image of the original clinical document object.

4.2.3. REMINE Anonymization Procedures

For protecting the patient privacy in the data mining and knowledge extraction process, the REMINE Anonymization Procedure was defined within the Consortium. A description of this procedure is provided in this subchapter.

Patients Personal Health Information (PHI) managed by ReMINE is taken in custody by the Hospital Information System hence anonymization is not required while data management is limited by HIS security network.

When electronic data collection systems were used and any PHI is going to leave the institution's custody (e.g., to be analyzed or mined) then anonymization is often required. The concern is that if data about the institution's patients was going into someone else's hands then it required additional protection.

In order to provide adequate coverage of practices for protecting EMR privacy and avoid any unauthorized use and disclosure of their Personal Health Information, ReMINE implemented an anonymization mechanism to safeguard PHI. This means remove any identifying information about the individual patients in the data set, hence making the re-identification of those individuals impossible.

Anonymization procedure is implemented by Knowledge Extraction and Data Mining component following these main scopes:

- De-personalization
- Limiting the data set collected

Data anonymization can be applied during 2 phases of the Knowledge Extraction and Data Mining task: collection, retention / persistence. These two activities are sequential in the study: data is collected, retained for the duration of the study.

If data is collected anonymously, then by definition it is anonymized during retention.

PHI may be collected but then removed soon after collection. That way the identifying information is collected temporarily but not stored in the Data Mining database. In this scenario it is important to determine which variables are identifying or potentially identifying so as not to collect them or to destroy them soon after collection. This practice is called limiting the data set. The Ethical Board provides guidance on which data will not be collected.

In order to keep the same permanent code for a given identity, so as to gather information for the same individual, ReMINE will assign a new aleatory Patient Identification Number (PID) to every HIS Patient ID. This new PID will be implemented in a single data anonymization session, it implies that subsequent session will generate different PID. As result of this technique, all sessions are non-incremental series.

The diagram below shows the workflow among these activities.

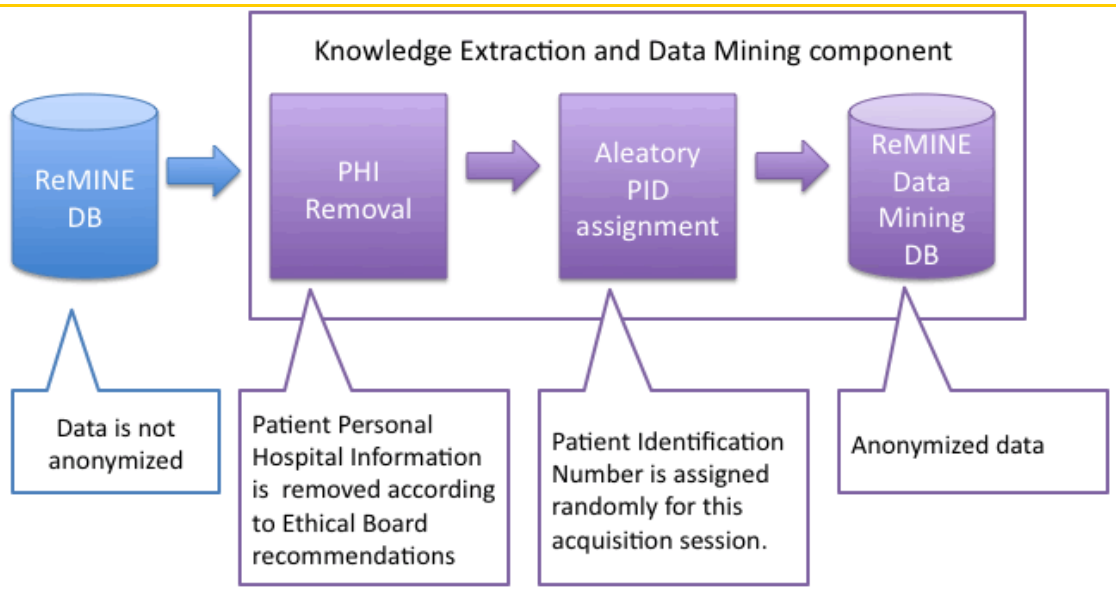

Figure 7. Workflow of data anonymization procedures

The technical process can be outlined as following: clinical documents are extracted from the RLUS repository where they are stored as binaries, are deserialized and stored in a temporary database. The extraction process is realized by a service named CDAService, as described in detail in the previous subchapter.

After the CDA documents are stored in the temporary database, data should be anonymized prior to the data mining process. For this, the patient-identifiable information was fully replaced by randomized patient IDs. Information such as the identification information of the patient's consulting physician was removed. Some IDs included by the ReMINE services in the CDA document were also removed for ensuring completely anonymized historical data. The final content obtained by this procedure represents the full set of minable data, without any Patient-identifiable Personal Health Information.

4.2.4. Data pre-processing

The data pre-processing step consist in an ETL process that extracts data from the general Data Mining database and, after a series of transformations, loads it into the REMINE_TRFT_DM database which contains the actual minable data.

4.2.4.1 Data selection

The process of selecting the data splits into two steps: selecting documents and selecting information from the document. Only the relevant CDA documents needed for the TRFT scenario must be selected and extracted from the Data Mining database. Moreover, only information directly implied into the mining process must be selected from the document itself.

The relevant clinical documents for the TRFT data mining scenario had to include anonymized patient information such as age and sex, hospital admission information such as hospital admission date, ward allocation and laboratory analysis information for establishing whether the patient was or not infected previously or during the hospital stay.

4.2.4.2 Data cleansing

The data cleansing step is critical for obtaining valuable mining results. Incorrect or duplicates values must be eliminated from the database. Records containing missing data should be appropriately handled. Incorrect values may be spotted by defining a valid range for the variable and eliminating values that occurs outside the range.

In the context of the TRFT scenarios, data cleansing included duplication identification and removal procedures, elimination of records containing missing data such as patient admission information without corresponding patient discharge information (due to the "window" over the data represented by the data in the dataset). Inconsistent values which are not in the normal, expected data range are also removed as they may corrupt the mining results.

4.2.4.3 Standardizing data

The standardization of data is mandatory for ensuring a common scale, when datasets for mining come from different sources, with different coding systems or different value ranges. In case of the data from the REMINE TRFT pilot, all data was already transformed to a common format and also a common coding system prior to being inserted in the RLUS metadabase.

4.2.4.4 Data transformation

The steps regards the activities of transforming data into new data records for suiting the data mining scenario modelling goals.The scenario aims at extracting association between patient health profile and the patient's hospital stay profile.

The **patient health profile** includes health information independent of the current hospital admission. The data for patient health profile includes the following variables:

- Patient demographic information such as age group and sex
- Background health information about the patient such as previous MRSA infections

Using useful prior domain knowledge, the patient health profile will include information about an age group instead of actual age, to provide a greater degree of generality to the profile.

The **hospital stay profile** consists in information regarding the hospital stay, the presence of other MRSA infections and, if the case, about a confirmed patient infection. The data for the hospital stay profiles includes the following list of variables:

- Existence of other infected patients in the same time (number of infected patients in the same ward with a certain patient)
- Available infection-related information such as laboratory results and observations which indicates a hospital-acquired MRSA infections, as the goal of the data mining scenario is to identify patterns that are used further to minimize the infection risk.
- The length of the patient's hospital stay.

The database diagram of the final REMINE_TRFT_DM database obtained after the ETL processing, by consolidating data from different CDA sections is given in the figure below.

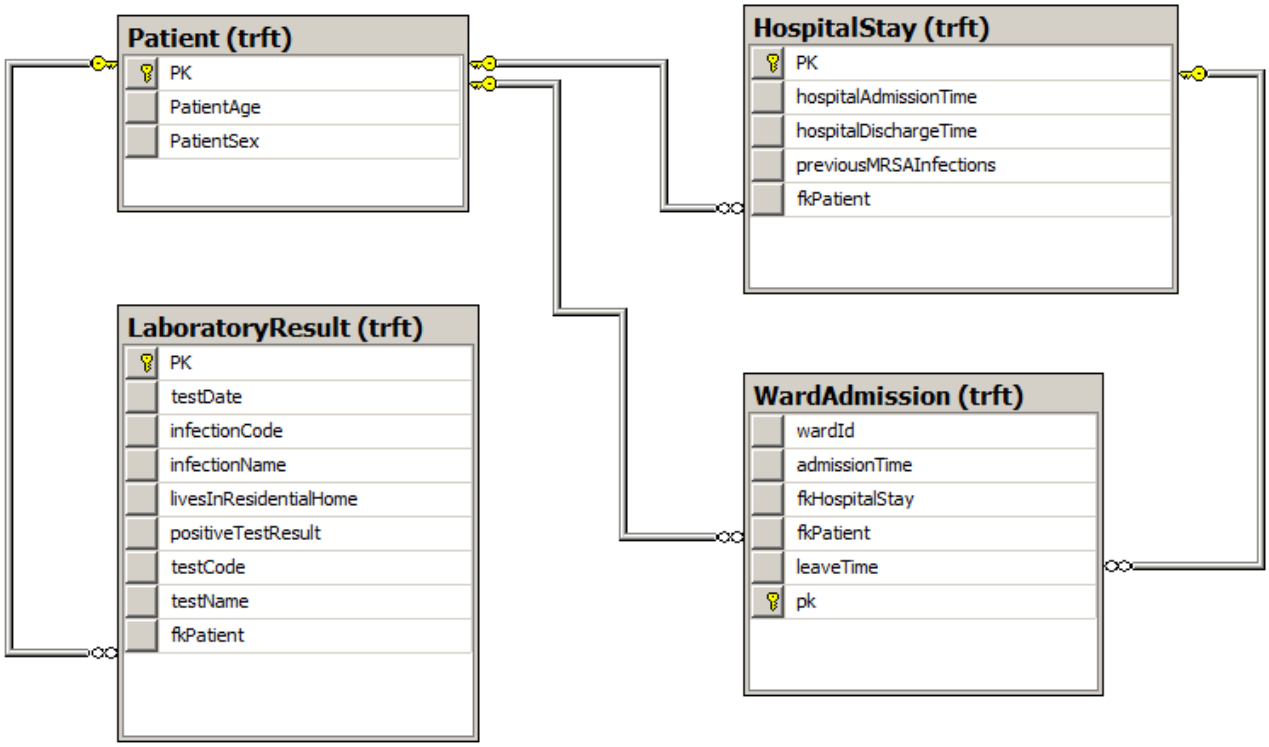

Figure 8. Database diagram for REMINE_TRFT_DM

4.2.5. Data mining modelling

The data mining modeling steps are performed using SQL Server Analysis Services, by developing a SSAS data mining project.

4.2.5.1 Building the Mining Structure

The data source of the mining project is configured to be the REMINE_TRFT_DM database described above.

The association mining model must contain a single Case Table and multiple Nested Tables. The mining approach suggests the use of case table and no nested table. Thus, the two types of profiles, patient health profile and the hospital stay profile, are constructed in a single table together. This is suitable also because the association between the two profiles is also modeled as 1-to-1, as the system generates a new pair of patient profile and hospital stay profile for each new patient admission.

The Mining Structure for the mining model including the constructed Case Table is presented below. The *PatientInfectionExposure* table includes both patient profile and hospital stay profile information.

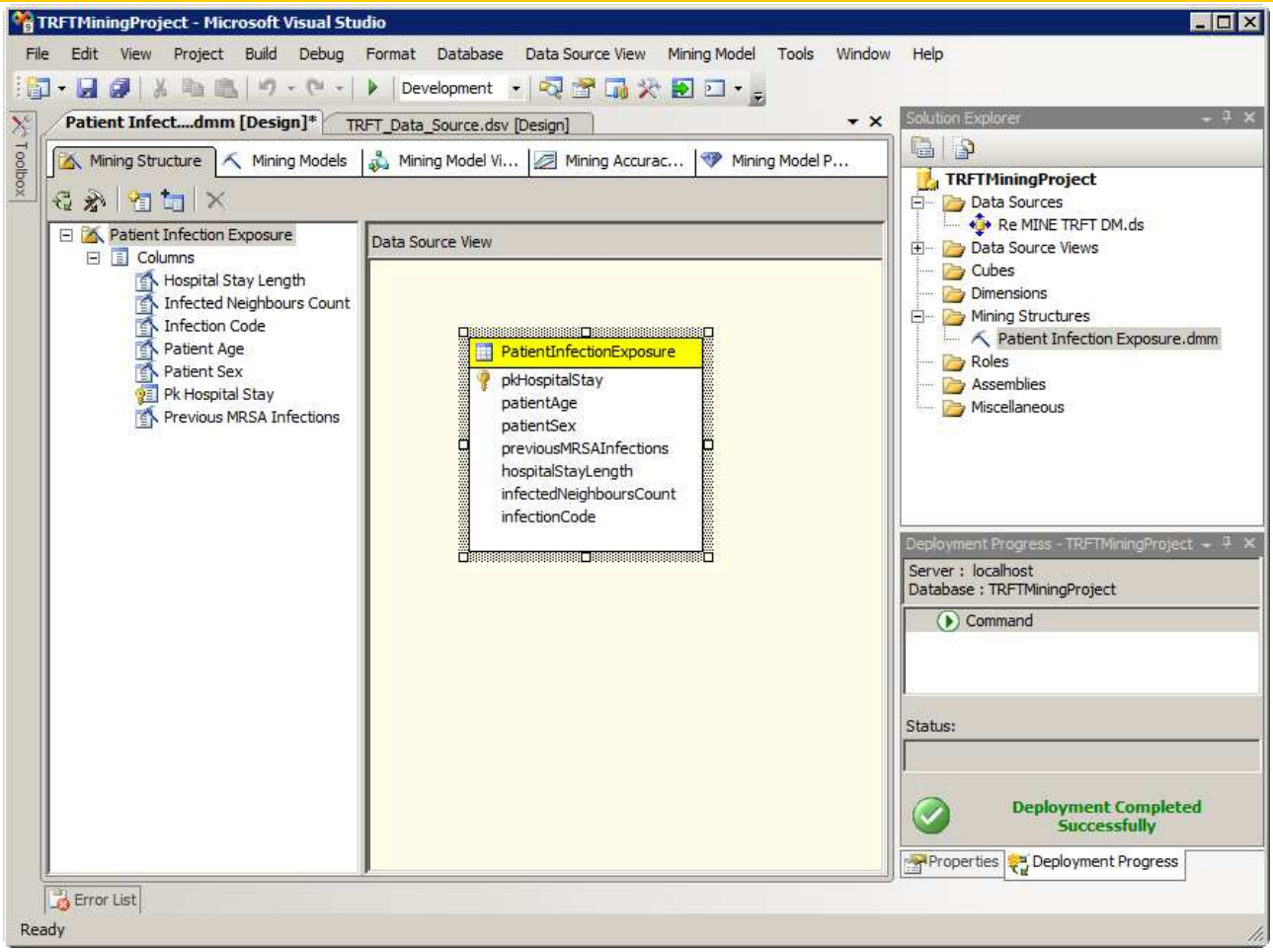

Figure 9. TRFT Mining Structure

4.2.5.2 Building the Mining Model

Mining models in Analysis Services require defining the role of each field in the model to be created. A field may be "predictable", "input" or "predict only", accordingly to the place it may occupy in the rule. Predictable fields may appear on both sides on the rule, input fields will appear only in the antecedent of the rule and predict-only fields may appear only in the consequent of the rule.

For the TRFT Mining Model, patient heath profile and hospital stay profile information will serve as input data, while infection code (which indicates a hospital-acquired MRSA infection) is specified as "predict-only" field. The figure below presents the mining model for the TRFT scenario.

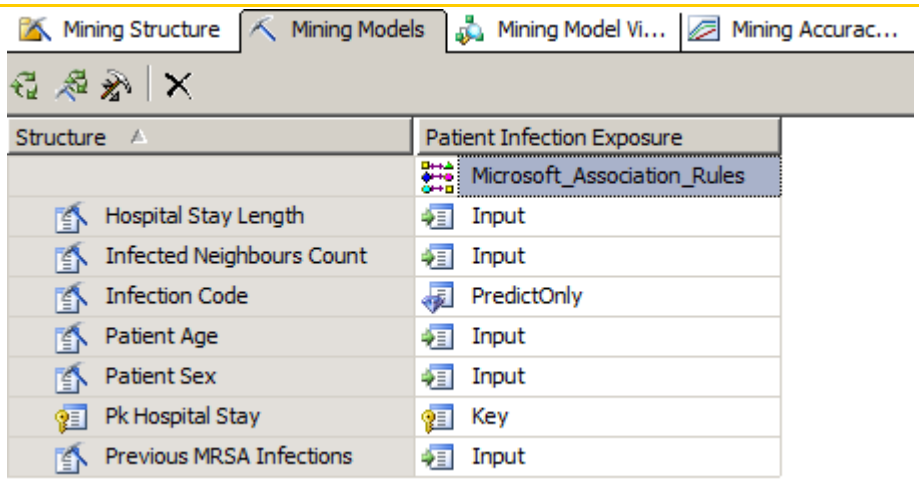

Figure 10. TRFT Mining Model

4.2.5.3 Parameterizing the Mining Algorithm

The Microsoft Association Rule Algorithm includes a set of parameters which governs the expressivity and complexity of the resulting mining model. The parameters that can be adjusted are the minimum and maximum support, the minimum and maximum confidence, minimum importance, and so on. Support measure controls how the rare items influence the data mining results, and how useful rules including these items are or not taken into consideration.

The parameters are established and re-tuned after the evaluation of mining results on a representative input sample of the data. The figure below presents the parameters we used.

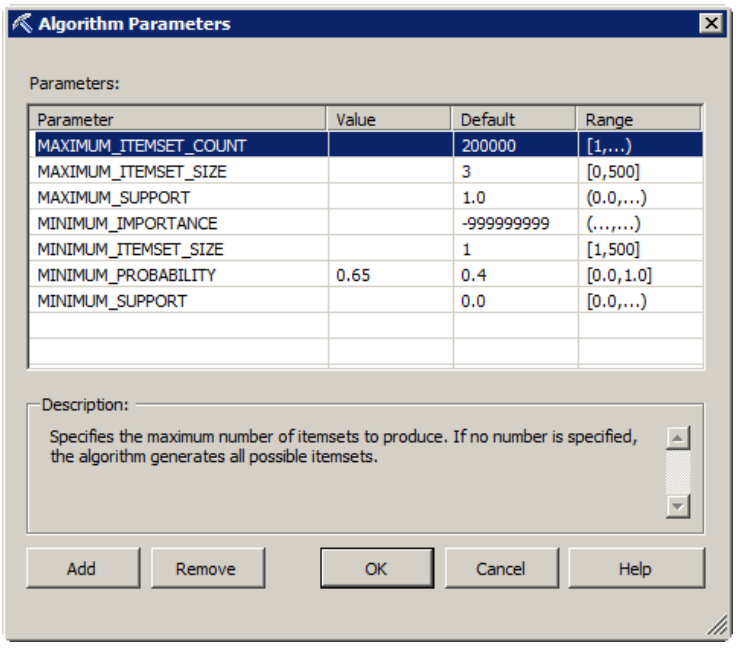

Figure 11. Parameters for the TRFT Association Rule Mining algorithm

4.2.5.4 The data mining process

The Analysis Services Mining Project developed in the previous steps is deployed on the SQL Server instance. The cases from the data source are processed and the output of the mining is stored into an Analysis Services database. The evaluation of the mining results may lead to modification of mining parameters, included mining variables or the choice of the mining algorithm itself. The mining process might be done periodically after a reasonable quantity of new data was collected, for improving mining pattern reliability.

4.2.5.5 Evaluation of results and retuning parameters

The association rules obtained from the mining process are evaluated for assessing how useful and actionable the mining patterns are. The evaluation can be a direct evaluation by domain experts or an indirect one based on evaluation the performances of the knowledge inference system using the mined rules. The evaluation in TRFT Mining Scenario was a critical step for producing quality mining models and led to important revisions of the included data to arrive to the present data source configuration.

4.3. Data mining results

The data mining process is finalized with an Analysis Services analytic database containing the association mining model. An association rule summarizes a pattern of association between the health profile of the patient combined with the patient hospital stay profile, and the presence of a MRSA infection. An aspect to be mentioned is that the association rule is a co-occurrence pattern, not a causal implication.

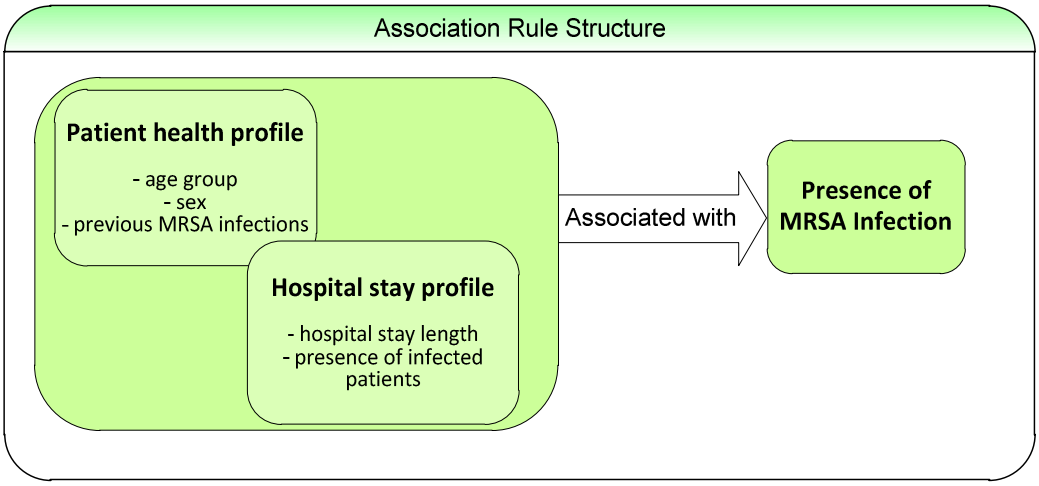

Figure 12. TRFT Scenario Association Rule Structure

The structure of the rule is presented into Figure 12. The two profiles on the left side of the rule consist in a patient health profile and a hospital stay profile. On the right hand, there is the presence of the MRSA Infection, expressed as a Boolean value in the underlying model. The rule expresses the co-occurence relationship between the two profiles and the possible contracting of the MRSA Infection in the hospital environment. Nor the patient health profile, neither the hospital stay profile contains actual patient data yet aggregated information. The mining algorithm performs automatic discretization in case of numerical attributes (hospital stay length, count of infected patients), the benefit of obtaining more general, actionable and reliable rules.

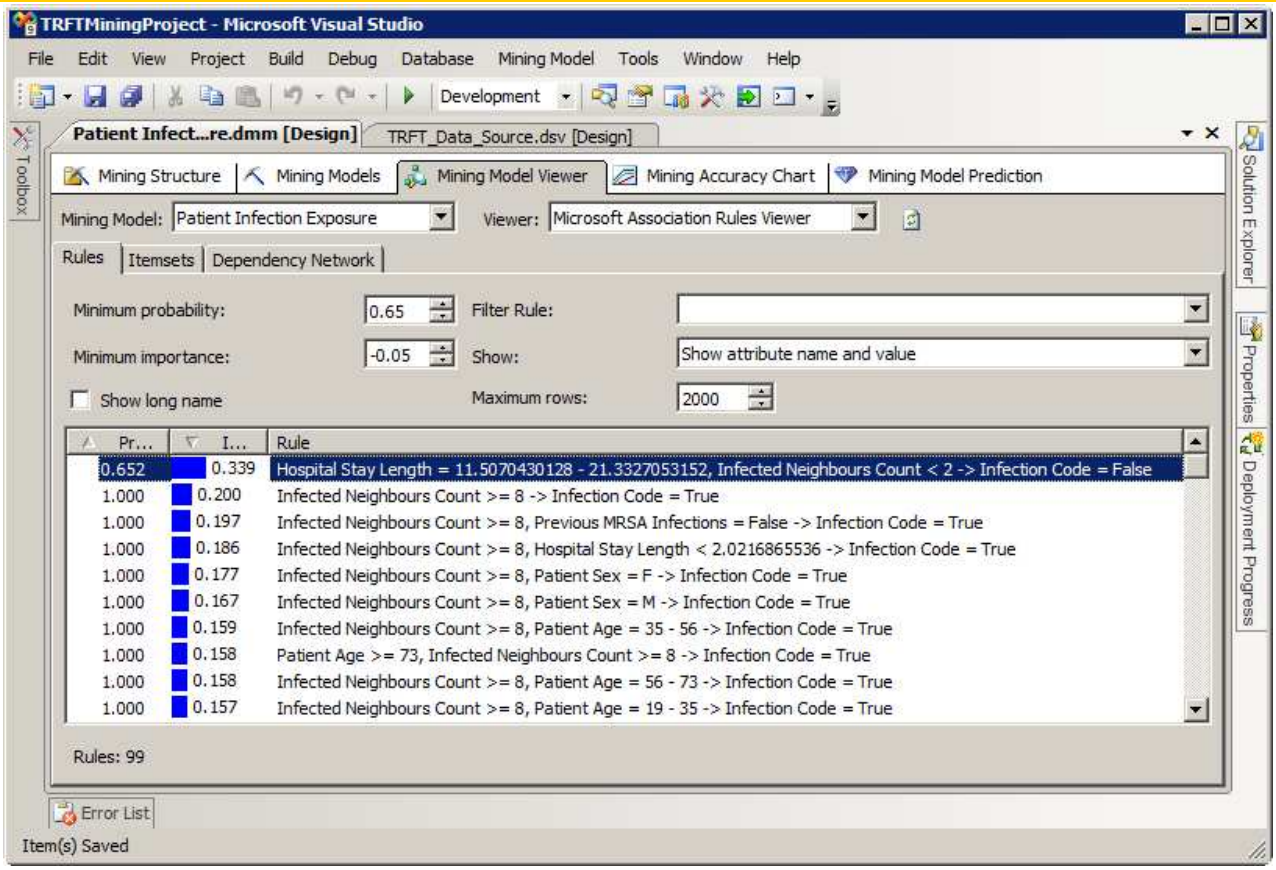

Figure 13. Output association rules from data minig process

Rules are useful in the detection process if they have the right level (not too general, not exceedingly specific) for not losing the actual relationship that exists between contracting the MRSA infection and the patient profile. The importance score of the rule is used to discriminate the most reliable rules and to prune the uninteresting rules when migrating rules to the knowledge base in the Knowledge Inference System.

5. Knowledge inference system enhancements

The Knowledge Inference System represents the component which provides real-time clinical information processing and identification of potential threats to patients' safety based on historical patterns. Thus, the data mining modules offers discovery capabilities within the captured historical patient data, based on several data mining scenarios as described previously in the preceding chapter and also in Deliverable D3.7 First Revision of the data mining and knowledge extraction module. In the following, we will overview the architecture and functioning of the Knowledge inference system and the interactions of the KIS component with other components within the DM & KE subsystem.

5.1. Overview of the knowledge inference system

The flow of the risk detection and alert delivery processes is described in the following figure. The central component which provides interoperation means for the Knowledge Inference System and the Risk Manager Interface (RMI) is represented by the WS Eventing Platform.

The KIS and RMI components register interest for certain WS Eventing notifications through the Subscription Manager Service. KIS registers a Push subscription which requests immediate notifications on the Document Receive Service, for the new clinical documents submitted to WS Eventing Platform. Each RMI client registers a Pull subscription targeting new KIS alerts. Both subscriptions contain filters which differentiate between document and alert messages, based on the data contracts exposed by RLUS and KIS.

RLUS Service submits new real-time documents about patient admission, ward transfers, laboratory results, etc, to the Dispatcher Service on the WS Eventing Platform. The Dispatcher Service will process the submitted messages and will create new notifications according to the already registered subscriptions. Notifications are stored in an internal message repository until they are delivered.

The Notification Service performs a periodic check of new messages in the internal repository and sees that a message for KIS through a Push subscription is available. Thus, the service immediately delivers the message to the Document Receive Service from KIS.

KIS performs the detection processes and submits the obtained alerts, if any, to the Dispatcher Service from WS Eventing. Again, the Dispatcher service creates and persists the appropriate messages based on the existing subscriptions. In this case, there is only a Pull subscription from RMI for the KIS alerts, thus the delivery will be performed on explicit demand from RMI.

The Risk Manager Interface is launched by the used on his computer. While running, RMI performs a periodical request for new alerts by the Pull Service from the WS Eventing Platform. When Pull Service is demanded for new alerts, it returns the messages awaiting delivery to the Alert Receive Service from RMI. The Alert Receive Service interacts with the Risk Manager User Interface for displaying the alerts to the end user.

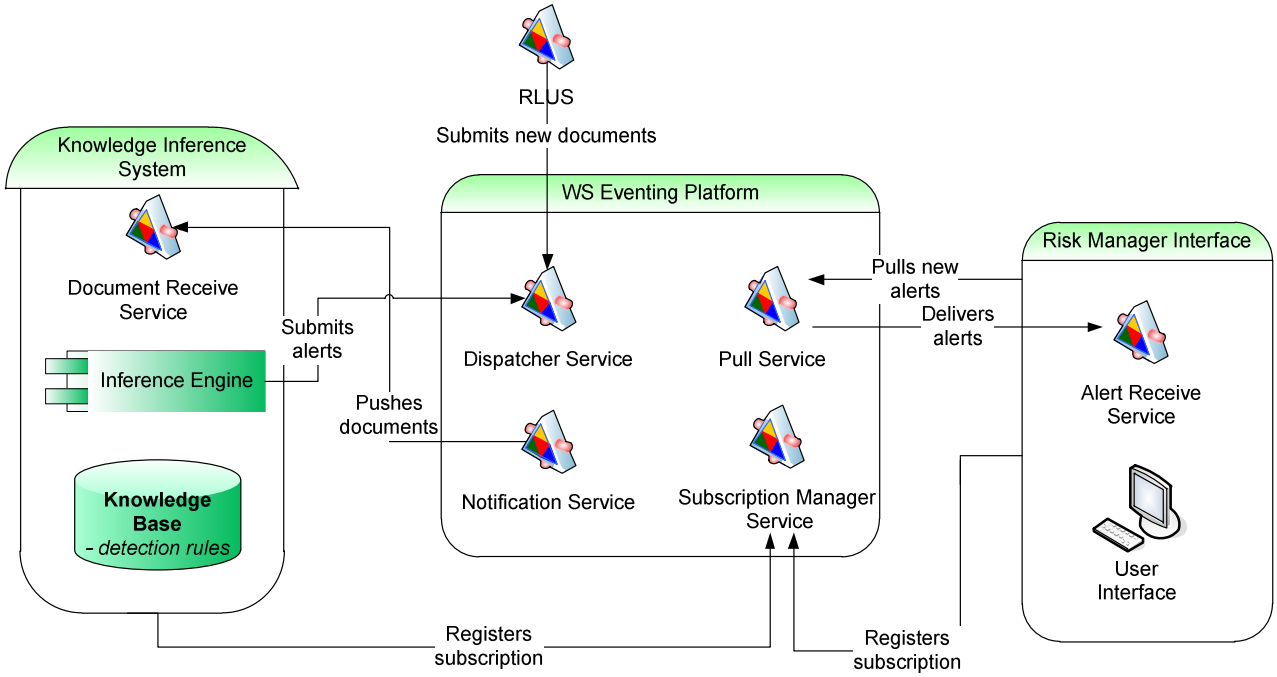

Figure 14. Overview of communication between RLUS, KIS and RMI components through the WS Eventing Platform

5.1.1. Integration of mined models

Based on the obtained mining models from the data mining scenarios, the structure and semantics of the rules to be used in the KIS were specified. One of the aspects considered in the data mining approach was the production of mining models that can be further used for performing risk prediction based on new data. Different data mining and machine learning approaches can be employed in a supervised learning process of risk patterns within the ReMINE context. The focus was to produce explainable detections of risk for providing the authorized medical personnel with in-depth, justified and reliable information about *why* some recorded patient data suggests the existence of health risk for the patient's health or for the other patients. Thus, the focus was to identify a data mining algorithm able to capture knowledge at a symbolic level, as opposed to sub-symbolic learning approaches such as artificial neural networks. The mining model obtained using this procedure could be subsequently processed and used as knowledge base for the artificial reasoning process in a business inference engine.

The association rule mining such as ***"Infected Neighbours Count >= N -> Infection Code = True"*** will be applied in the KIS in such way that when a patient with a high number of infected neighbor patients in same ward (hospital section) – greater than the N discovered constant - is identified, a risk alert will be sent to the infection control team manager indicating the fact that the patient may contract the infection with a high probability (computed from the rule importance).

The Knowledge Migration component was developed to perform easy integration of the mined models with the Knowledge Base of the Inference Engine. The mined patterns are stored in an analytical database and should be transformed to the data model for rules and facts used in the KIS system. The access to the Mining Model in the SQL Server Analysis Services is performed by using the ADOMD / AMO libraries. The

ADOMD library offers an easy access to the contents to the data mining model. The knowledge migration component extracts the association mining rules, transforms them to the adopted data model and loads them in the Inference Engine knowledge base.

5.1.2. KIS Architecture Overview

In this subchapter, the internal architecture of Knowledge Inference System will be presented. The components involved for KIS are targeted at two goals:

1) Producing risk detections based on patterns and new incoming data
2) Ensuring interactivity with the other DM & KE subcomponents through the use of the message routing capabilities of the WS Eventing Platform.

Below, the architecture of the Knowledge Inference System is outlined, including internal subcomponents and the KIS interconnection with external components in the DM & KE system.

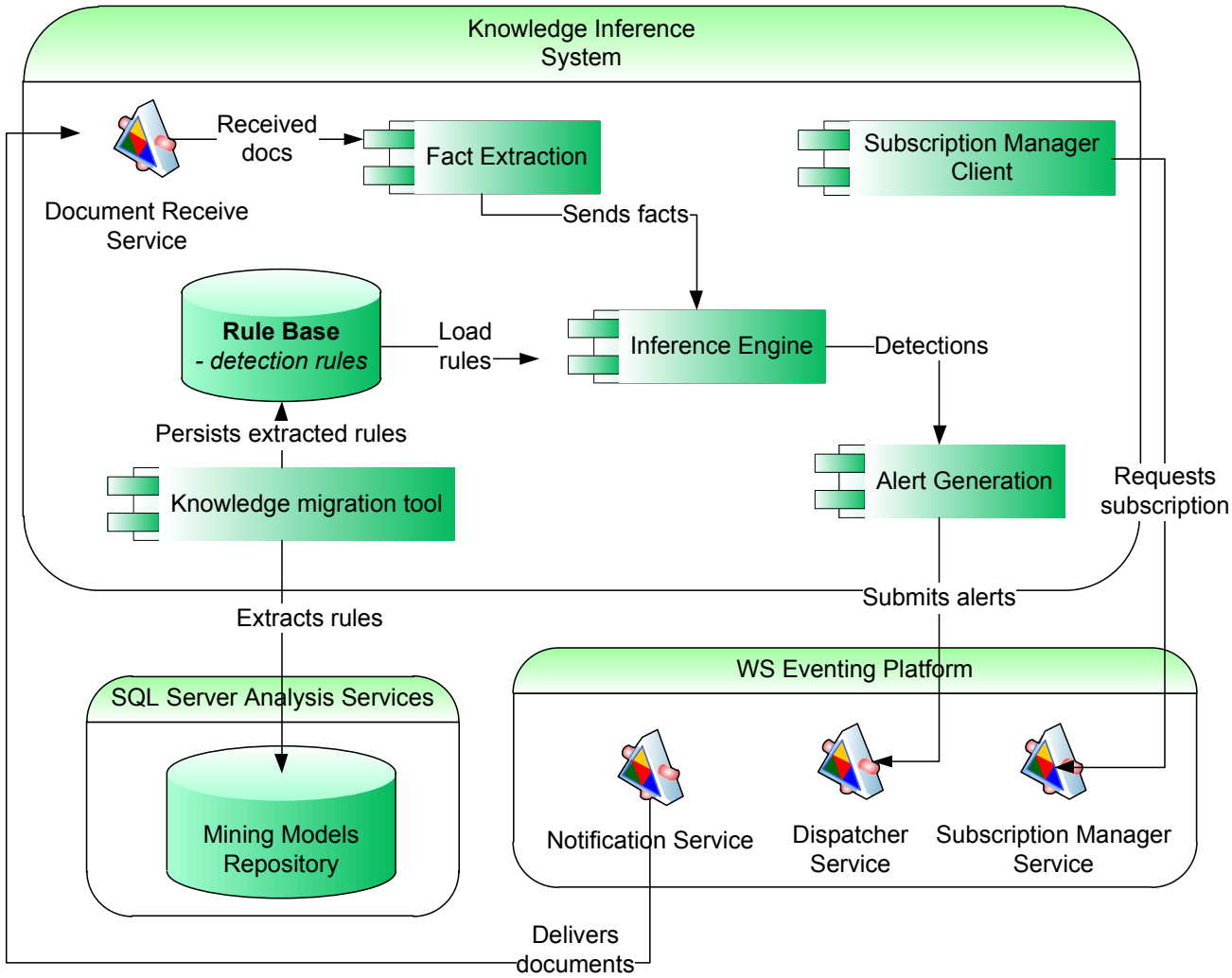

Figure 15. Architectural overview of KIS

The Knowledge Inference System receives real-time new clinical documents resources from the WS Eventing Platform and performs the necessary risk detection algorithms based on the rules discovered through data mining.

For ensuring the connection with WS Eventing Platform, KIS has a Subscription Manager Client component which supports the registration of the Push subscription for clinical documents to the Subscription Manager Service. The subscription includes a XPath filter which identifies messages containing the types of clinical documents targeted by KIS.

The notifications are delivered by the WS Eventing Notification Service to the Document Receive Service, part of the KIS' external interface. Notifications contain the new resource - which consists in a binary payload (a serialized HL7 CDA instance) and metadata information about the CDA instance – and also the identifier of the operation which was executed by RLUS and generated the notification, such as to create, delete or update the resource.

The **Fact Extractor** component receives the Clinical Document object and performs a set of queries for extracting the relevant information in risk detection. The information is dependent by what categories of data are checked by the rules loaded in KIS. Rules are dependent, in turn, by the mining scenarios from which they originate. Thus, the Fact Extractor is incrementally modified for extracting information which can be validated by means of the patterns derived from data mining. The adoption of new mining scenarios needs small modification at the level of the Fact Extractor. The facts, which are plain C# object according to a designated data model, are fed further to the Inference Engine.

The **Inference Engine** is the core component of KIS and performs the reasoning processes in this subsystem. The Engine employs, on one hand, **facts -** which express the current state of the evaluated situation and, on the other hand, **rules** – which express the knowledge patterns that are used to derive new facts. Facts are represented in the KIS system by the variables extracted from clinical documents, which give the overview of the situation of a patient, while rules are obtained from the mined patterns and are used to derive contradictions between the current facts and normal/low-risk patterns as well as to match facts with high risk patterns. More information on how inference engine is structured and functioning is presented in the next subchapter.

The **Alert Generation** component identifies from the facts derived in the process of applying rules against facts, certain types of facts that represent detections/alerts. The alert generation component collects the alerts from the fact memory of the Inference Engine, includes any necessary additional information for the alert object and builds the final alert message. The component also handles the transfer of the alert to the WS Eventing Platform, by including a client implementation for the WS Eventing Dispatcher Service. As KIS submits the alerts to the WS Eventing platform, the WS Eventing prepares the alert messages to the RMI clients accordingly to the existing Pull subscriptions.

The **Knowledge Migration Tool** ensures the extraction and transformation of the mined patterns from the Mining Model Database in the SQL Server Analysis Services. The migration process was previously documented in the subchapter 5.1.1.

5.1.3. KIS Inference Engine

The main goal of the KIS Inference Engine is to provide means for applying the rule-based mined models against new, unseen data and predicting the risk level associated with a patient situation.

As opposed to traditional procedural control, the inference engine brings a different information paradigm, the data-driven computation, and blends it with the implementation of simple artificial reasoning methods.

The engine employs as the unit of declarative knowledge the *rule*, which expresses the conditions that must be true for a certain action to become eligible for execution. However, rules differs a lot from traditional "if-then" approach. Rules are independent units which, depending on the input data (i.e the *facts*), can be executed or not. As the execution of a rule may change the current facts, new rules can become applicable or cease to be applicable.

There are two main types of reasoning when using rules. *Forward chaining* starts with the available data and uses inference rules such as from applicable rules (where the antecedent holds); more valid data is derived from the rule consequent. The "chaining" of inferring new data based on rules continues until a goal is reached. *Backward chaining* starts with a list of goals that must be reached and works backwards from the consequent of the rule to the antecedent to see if there is data available that will support any of these consequents. The KIS Inference engine employs forward chaining, starting with available variables with patient information and subsequently applying rules against them.

The main components of the KIS Inference Engine component are outlined in the following:

- **Knowledge Base** – stores the set of rules derived from data mining models
- **Working Memory** – contains the facts, which are variables extracted by the Fact Extractor components from the CDA objects delivered to the KIS component
- **Inference Engine** – the core component that encapsulate the inference logic; it contains:
    - a **Pattern Matcher** module – performs matching of the facts with the rules they enable; a system with a large number of rules and facts may result in many rules being true for the same fact assertion
    - an **Agenda** – manages the execution order of these conflicting rules using a Conflict Resolution strategy, which is represented here by the *rule importance*, a specific measure of the interestingness of an association rule computed in the SQL Server Association Rule Mining Algorithm.

The **functioning cycle** implemented in the KIS Inference Engine matches with the typical functioning cycle for a forward chaining business rules engine and be can be split in three phases, as detailed below.

Matching – facts (CDA objects) introduced to the engine are matched against rule conditions (conditions modeling risk profiles obtained from data mining). The rule engine determines which rules are applicable based on the given facts, the input required for the rule, as well as the results obtained from the evaluation of previous rules. If a fact matches a rule's condition, then that rule becomes a candidate for execution and is added to the rules engine agenda. All facts that match the active policy rules are inserted in the agenda at the end of this phase.

Conflict Resolution – the candidate rules on an agenda are analyzed and compared to determine if any execution sequencing conflicts exist in the logic of the rules. If any are determined, the conflicts are resolved according to the execution sequence strategy that is defined by the user. One common conflict resolution strategy is based on rule priority, and in this case the engine sorts the agenda based on priority (high to low). The rule with the highest priority will be executed in the next phase.

Execution – the actions defined in the "action" component of the selected rule are executed. Actions can assert new facts or retract/update facts into the rule engine, which causes the cycle to continue. This is known as forward chaining –where the match-resolve-act cycle repeats for rules affected by a changing fact

base. From the execution phase, a set of special facts which represents detections / raw alerts are produces and stored in the working memory.

After the inference engine execution has completed the processing of a given set of facts, the alert facts from the working memory are selected and further processed by the alert generation component, which eventually dispatches them to the WS Eventing Platform.

5.1.4. Service contracts

The API for KIS consists of three components: service contracts, data contracts and fault contracts, according to the model defined by the .NET WCF (Windows Communication Foundation). Contracts are used to standardize the communication between the client and the service. Different contract types define operations that can be performed, data types that are transferred and errors that are propagated between the service and the client.

- Service contracts – describe the operations offered by the service
- Data contracts – describe the data types that are exchanged between the service and the client. The data contract describes how parameters and return types are serialized into XML in order to be transferred. The built-in types have implicit contracts, but custom types needs explicit data contracts.
- Fault contracts – describe how errors that are raised in the service are communicated to the client.

KIS exposes one run-time interface that can be accessed by the Risk Manager Interface in order to check for pending risk notifications and to retrieve them, in the case that waiting notifications are found. Risk notifications will be derived from the CDA documents that KIS evaluates and validates by using the knowledge obtained through data mining.

Internally, DEMS will notify KIS of new data. KIS validates each new clinical document by applying the appropriate rules from the knowledge base. If the risk is over a threshold level, an alert notification is compiled with information about the risky parameters, the risk severity and the source clinical document. If no alert can be inferred, a null is returned.

5.1.4.1 Document Receive Service Interface

In the following, the contracts for the Document Receive Service are described.

Service Contract

- Notification

Business friendly name	Receives new clinical document notifications
Signature	`void Notification(string operationId, string resourceId, Resource resource);`
Description	The operation is used for receiving the newly arrived clinical documents

	(HL7 CDA) which should be evaluated for detecting risk. The Notification Service from the WS Eventing Platform calls this operation for delivering the new documents in a push manner (immediately when received), submitted by an event source. The original event source is this case it the Retrieve Locate Update Service. The received documents are parsed and fed to the Fact Extraction component for obtaining the raw facts to push into the inference engine.
Inputs	operationId - The identifier of the operation executed on the resource (Created, Deleted, Updated) resourceId - The identifier of the resource resource – The new resource including the CDA document to be processed
Outputs	None
Exceptions	*InvalidOperationId* *ResourceFormatException* *UnexpectedError* *TimeoutError*

Data contracts

- **Resource**

Class Description	Represents the RLUS data type containing information about the resource which is delivered to KIS.
Data Members	- Metadata about the resource - Resource Binary

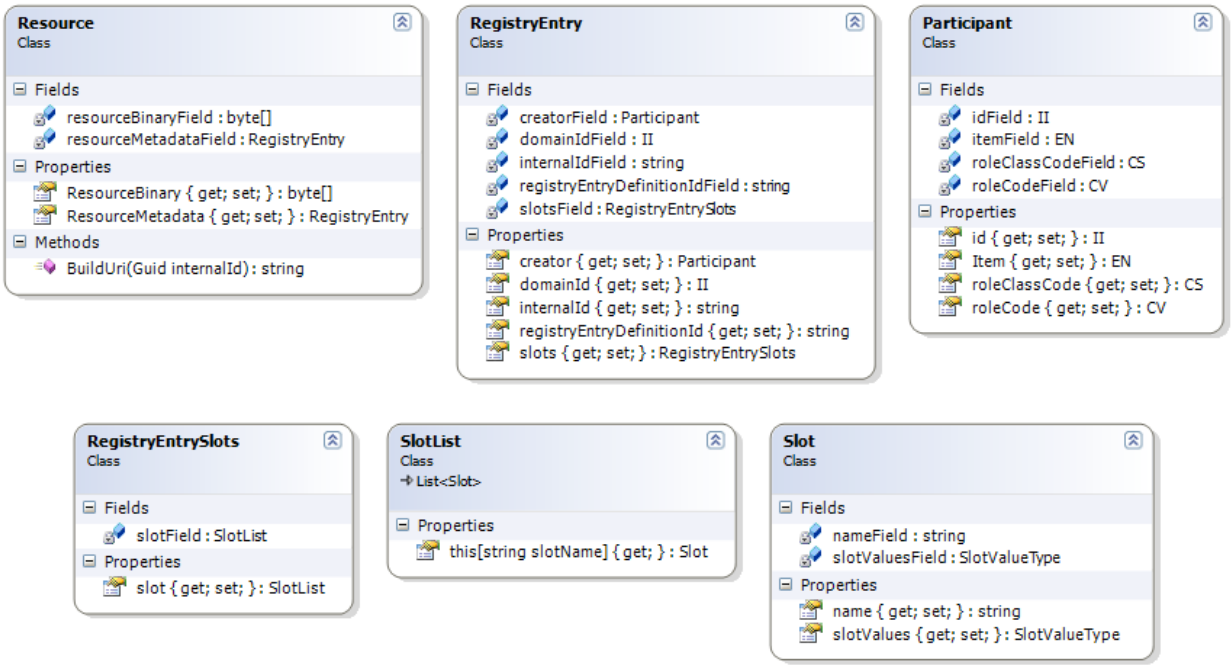

Figure 16. Class diagrams related to the RLUS Resource data type

Fault contracts

Class	Description
InvalidOperationId	Size is negative or zero.
ResourceFormatException	The provided resource binary is in an illegal format and cannot be deserialized.
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.
TimeoutError	The time limit configured for the API service to produce a result has expired

5.1.4.2 Alerts data contract

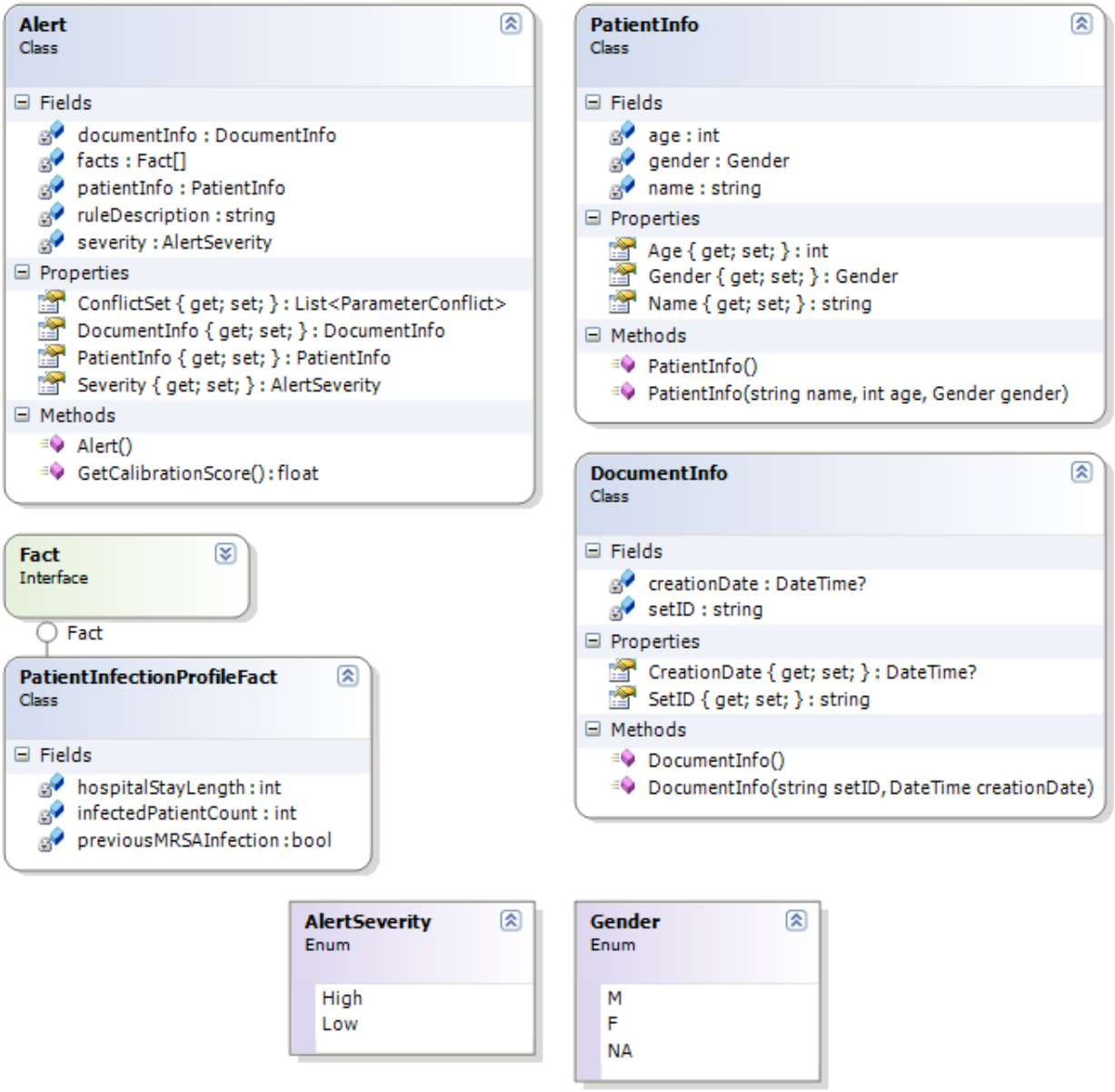

Figure 17. Data contracts for the Alert objects submitted by KIS to WS Eventing Platform

5.2. Risk Manager Interface

The Risk Manager Interface (RMI) represents a part of the RAPS management system which facilitates the visualization of the identified risks and supports the activities of the designated risk managers in the hospital. RMI consists in a client application which is installed on the end users' computers and runs in the background as a Windows Tray application along with a Windows Service for receiving notifications and additional client endpoints for accessing WS Eventing services. The main goal of the RMI is to provide real-time alerting capabilities to the risk managers by employing interactive notifications on their computers' desktop.

5.2.1. Risk Manager Interface Architecture Overview

An overview of the Risk Manager Interface component architecture and interactions with other related systems is detailed below, based on the Figure 18.

The Risk Manager Interface is composed from the main User Interface, which ensures the visual means for providing decision support to the designated risk managers, and the components which support the connectivity with the WS Eventing Platform for alerts retrieval.

The RMI client registers a Pull subscription to the WS Eventing platform for the targeted alerts through a Subscription Manager Client component. The subscription specifies which types of alerts are interesting for the risk manager, such as only for alerts regarding a set of wards for which the risk manager is responsible for. The alert retrieval is performed by a Pull Receive Worker component, which performs periodical calls for new messages to the Pull Service on the WS Eventing Platform. The obtained new messages are send on a RMI internal interface, the Alert Receive Service, for decoupling the message-checking mechanisms from the handling activities when new alerts are received. The Alert Receive Services performs any necessary message transformations and sends the alerts to the User Interface for being presented to the end user.

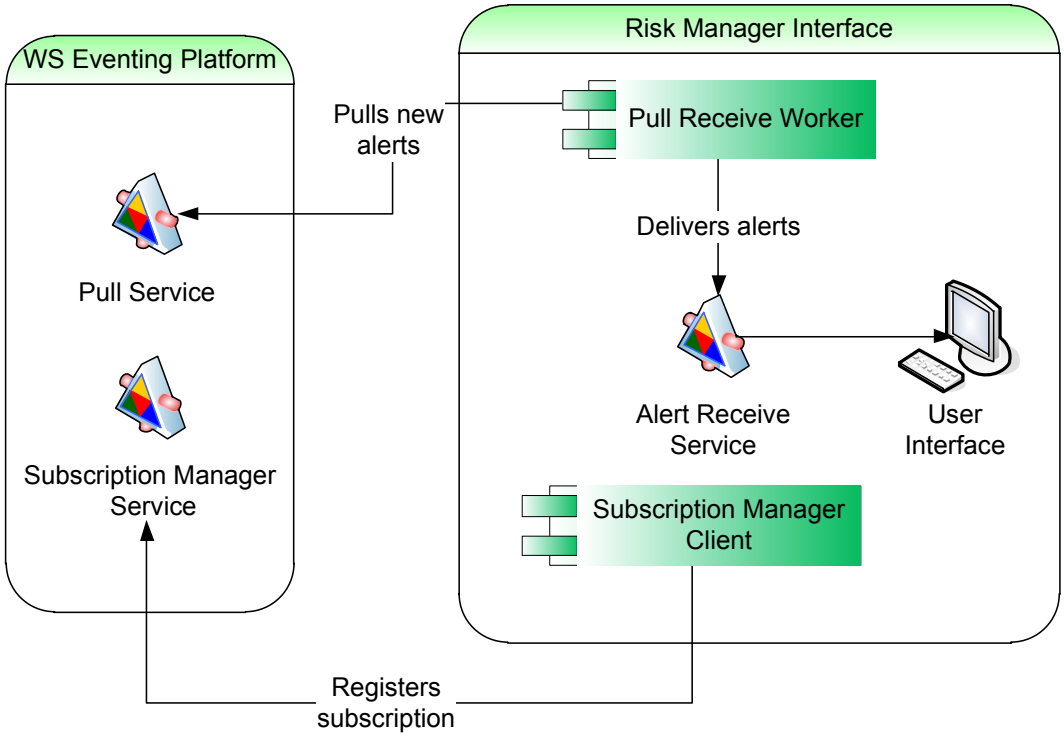

Figure 18. RMI components overview

5.2.2. Service Contracts

The API for RMI consists of three components: service contracts, data contracts and fault contracts, according to the model defined by the .NET WCF (Windows Communication Foundation). Contracts are used to standardize the communication between the client and the service.

RMI employs the PullReceiveWorker, a component included in the WS Eventing library, in order to periodically check for pending risk notifications and to retrieve them, in the case that waiting notifications are found. RMI exposes one run-time interface, Alert Receive Service, which is accessed by the Pull Receive Worker for delivering the obtained alerts. Internally, Alert Receive Service will process the incoming alerts and send to the User Interface for being displayed to the end user.

5.2.2.1 Alert Receive Service Interface

In the following, the contracts for the Alert Receive Service are described.

Service Contract

- Notification

Business friendly name	Receives the risk alerts awaiting for delivery
Signature	`void Notification(Alert alert);`
Description	The operation is used for receiving the alerts describing the risk indicated by newly arrived and evaluated clinical documents. The PullReceiveWorker component calls this operation for delivering the obtained alerts from the WS Eventing Pull Service. The alerts are created by the KIS risk detection process on the new clinical documents and submitted by KIS to the WS Eventing Dispatcher Service.
Inputs	The alert to be delivered to the RMI component.
Outputs	None
Exceptions	*AlertFormatException* *UnexpectedError* *TimeoutError*

Data contracts

- **Alert**

Class Description	Represents the data type containing information about the alert that is produced by KIS and presented by the Risk Manager Interface.
Data Members	- identification information about the patient

	- the clinical context - reference to the actual patient data that raised the risk suspicion (i.e. the risky parameters) - data that quantifies the risk and (i.e. individual risk parameter severity indicators and/or a global risk severity indicator) - additional information needed in the UI component (display names for infection codes and so forth)

Fault contracts

Class	Description
AlertFormatException	The provided alert has fields with illegal values and thus is unusable.
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.
TimeoutError	The time limit configured for the API service to produce a result has expired

5.2.3. Risk Manager Interface Engineering

The Risk Manager Interface is comprised of a User Interface part and the core features which support the process of obtaining alert notifications from the Eventing Platform. The User Interface part was specifically designed after a careful analysis of the characteristics of the users, their needs and the activities they perform. The Risk Manager Interface is designed to be run in the Windows System Tray from the risk manager's computer startup for ensuring the alerts are continuously and timely delivered to the user computer.

An important reference in the user interface engineering was the Microsoft Health Common User Interface Guidelines (MSCUI)[1], which provide guidance for performing safe user interface design targeted at the healthcare environment. Bearing in mind the design guidelines proposed by MSCUI, the final User Interface of the RMI component had to be easy to use, had to feature the right level of detail for the risk managers and provide adequate means for visually delivering alerts from different priority classes.

[1] Microsoft Health Common User Interface - http://www.mscui.net/

The implementation of the RMI component was performed using Microsoft Windows Presentation Foundation technology for the User Interface and using the Windows Communication Foundation libraries for the interaction with the web services within the Eventing platform.

The following two CDA documents were processed by the KIS component and, based on information about the other patients in the ward, raised the following alerts in Risk Manager Interface.

CDA Document 1:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<ClinicalDocument xmlns="urn:hl7-org:v3"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
  <typeId extension="POCD_HD000040" root="2.16.840.1.113883.1.3" />
  <templateId root="2.16.840.1.113883.3.27.1776" />
  <id extension="1234567-12345" root="2.16.840.1.113883.3.270.1.1" />
  <code code="51855-5" codeSystem="2.16.840.1.113883.6.1"
    codeSystemName="LOINC" displayName="Clinical note" />
  <effectiveTime value="20100710000000" />
  <confidentialityCode code="N" codeSystem="2.16.840.1.113883.5.25"
    codeSystemName="Confidentiality" displayName="Normal" />
  <setId extension="AA1234" root="2.16.840.1.113883.3.270.1.2" />
  <versionNumber value="1" />
  <recordTarget>
    <patientRole>
      <id extension="AA1234" root="2.16.840.1.113883.3.270.1.3" />
      <addr>
        <streetName>STREET NAME</streetName>
        <city>CITY NAME</city>
      </addr>
      <patient>
        <name>
          <given>John</given>
          <family>Doe</family>
        </name>
        <administrativeGenderCode code="M"
          codeSystem="2.16.840.1.113883.5.1" codeSystemName="AdministrativeGender"
          displayName="Male" />
        <birthTime value="19750113" />
      </patient>
    </patientRole>
  </recordTarget>
  <author>
    <time value="20100710000000" />
    <assignedAuthor>
      <id extension="TC-PAS" root="2.16.840.1.113883.3.270" />
      <representedOrganization>
        <id root="2.16.840.1.113883.3.270" />
      </representedOrganization>
    </assignedAuthor>
  </author>
  <custodian>
    <assignedCustodian>
      <representedCustodianOrganization>
        <id root="2.16.840.1.113883.3.270" />
        <name>TC</name>
      </representedCustodianOrganization>
    </assignedCustodian>
  </custodian>
  <componentOf>
```

```
    <encompassingEncounter>
      <code code="338709012" codeSystem="2.16.840.1.113883.6.96"
        codeSystemName="SNOMED-CT" displayName="Triage" />
      <effectiveTime value="20100709" />
    </encompassingEncounter>
  </componentOf>
  <component>
    <structuredBody>
      <component>
        <section>
          <code code="29554-3" codeSystem="2.16.840.1.113883.6.1"
            codeSystemName="LOINC" displayName="Procedure" />
          <title>Patient Admission</title>
          <entry>
            <procedure classCode="PROC" moodCode="EVN">
              <code code="32485007" codeSystem="2.16.840.1.113883.6.96"
                codeSystemName="SNOMED-CT" displayName="Hospital admission" />
              <statusCode code="completed" />
              <effectiveTime value="20100625" />
            </procedure>
          </entry>
        </section>
      </component>
      <component>
        <section>
          <id extension="CARD" root="2.16.840.1.113883.3.270" />
          <code code="225746001" codeSystem="2.16.840.1.113883.6.96"
            codeSystemName="SNOMED CT" displayName="Ward">
            <qualifier>
              <name code="410511007" codeSystem="2.16.840.1.113883.6.96"
                codeSystemName="SNOMED CT" displayName="Current or past" />
              <value code="410513005" codeSystem="2.16.840.1.113883.6.96"
                codeSystemName="SNOMED CT" displayName="Past" />
            </qualifier>
          </code>
          <title>Admission to ward</title>
          <entry>
            <procedure classCode="PROC" moodCode="EVN">
              <code code="305342007" codeSystem="2.16.840.1.113883.6.96"
                codeSystemName="SNOMED-CT" displayName="Admission to ward" />
              <statusCode code="completed" />
              <effectiveTime value="20100625" />
            </procedure>
          </entry>
        </section>
      </component>
    </structuredBody>
  </component>
</ClinicalDocument>
```

CDA Document 2:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<ClinicalDocument xmlns="urn:hl7-org:v3"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
  <typeId extension="POCD_HD000040" root="2.16.840.1.113883.1.3" />
  <templateId root="2.16.840.1.113883.3.27.1776" />
  <id extension="1234567-12345" root="2.16.840.1.113883.3.270.1.1" />
```

```
<code code="51855-5" codeSystem="2.16.840.1.113883.6.1"
  codeSystemName="LOINC" displayName="Clinical note" />
<effectiveTime value="20100711000000" />
<confidentialityCode code="N" codeSystem="2.16.840.1.113883.5.25"
  codeSystemName="Confidentiality" displayName="Normal" />
<setId extension="AB4531" root="2.16.840.1.113883.3.270.1.2" />
<versionNumber value="1" />
<recordTarget>
  <patientRole>
    <id extension="AB4531" root="2.16.840.1.113883.3.270.1.3" />
    <addr>
      <streetName>STREET NAME</streetName>
      <city>CITY NAME</city>
    </addr>
    <patient>
      <name>
        <given>Jane</given>
        <family>Doe</family>
      </name>
      <administrativeGenderCode code="F"
        codeSystem="2.16.840.1.113883.5.1" codeSystemName="AdministrativeGender"
        displayName="Female" />
      <birthTime value="19450213" />
    </patient>
  </patientRole>
</recordTarget>
<author>
  <time value="20100711000000" />
  <assignedAuthor>
    <id extension="TC-PAS" root="2.16.840.1.113883.3.270" />
    <representedOrganization>
      <id root="2.16.840.1.113883.3.270" />
    </representedOrganization>
  </assignedAuthor>
</author>
<custodian>
  <assignedCustodian>
    <representedCustodianOrganization>
      <id root="2.16.840.1.113883.3.270" />
      <name>TC</name>
    </representedCustodianOrganization>
  </assignedCustodian>
</custodian>
<componentOf>
  <encompassingEncounter>
    <code code="338709012" codeSystem="2.16.840.1.113883.6.96"
      codeSystemName="SNOMED-CT" displayName="Triage" />
    <effectiveTime value="20100709" />
  </encompassingEncounter>
</componentOf>
<component>
  <structuredBody>
    <component>
      <section>
        <code code="29554-3" codeSystem="2.16.840.1.113883.6.1"
          codeSystemName="LOINC" displayName="Procedure" />
        <title>Patient Admission</title>
        <entry>
          <procedure classCode="PROC" moodCode="EVN">
             <code code="32485007" codeSystem="2.16.840.1.113883.6.96"
                    codeSystemName="SNOMED-CT" displayName="Hospital admission" />
             <statusCode code="completed" />
```

```
            <effectiveTime value="20100625" />
          </procedure>
        </entry>
      </section>
    </component>
    <component>
      <section>
        <id extension="CARD" root="2.16.840.1.113883.3.270" />
        <code code="225746001" codeSystem="2.16.840.1.113883.6.96"
          codeSystemName="SNOMED CT" displayName="Ward">
          <qualifier>
            <name code="410511007" codeSystem="2.16.840.1.113883.6.96"
                  codeSystemName="SNOMED CT" displayName="Current or past" />
            <value code="410513005" codeSystem="2.16.840.1.113883.6.96"
                  codeSystemName="SNOMED CT" displayName="Past" />
          </qualifier>
        </code>
        <title>Admission to ward</title>
        <entry>
          <procedure classCode="PROC" moodCode="EVN">
            <code code="305342007" codeSystem="2.16.840.1.113883.6.96"
                  codeSystemName="SNOMED-CT" displayName="Admission to ward" />
            <statusCode code="completed" />
            <effectiveTime value="20100625" />
          </procedure>
        </entry>
      </section>
    </component>
  </structuredBody>
 </component>
</ClinicalDocument>
```

In the following, the alert display with the Risk Manager Interface is illustrated with a set of screenshots and the associated CDA documents which raised the alerts.

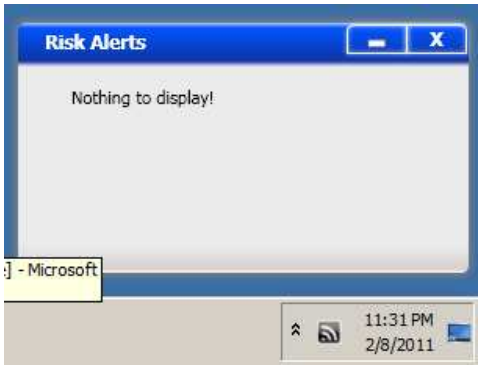

Figure 19. Appearance of the Risk Manager Interface with no alerts

The screen captures for the Risk Manager Interface including the alerts associated with these two documents:

Figure 20. Risk Manager Interface. Appearance of the popup window with 2 alerts

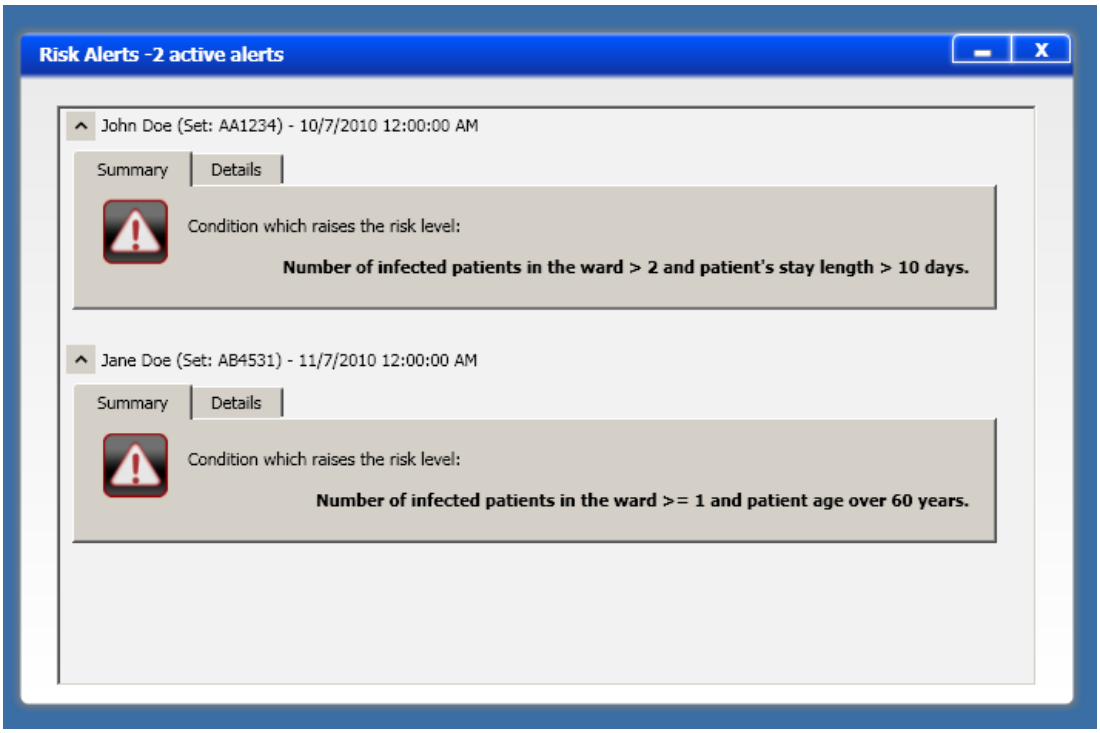

Figure 21. Risk Manager Interface. Appearance of the full window with 2 alerts

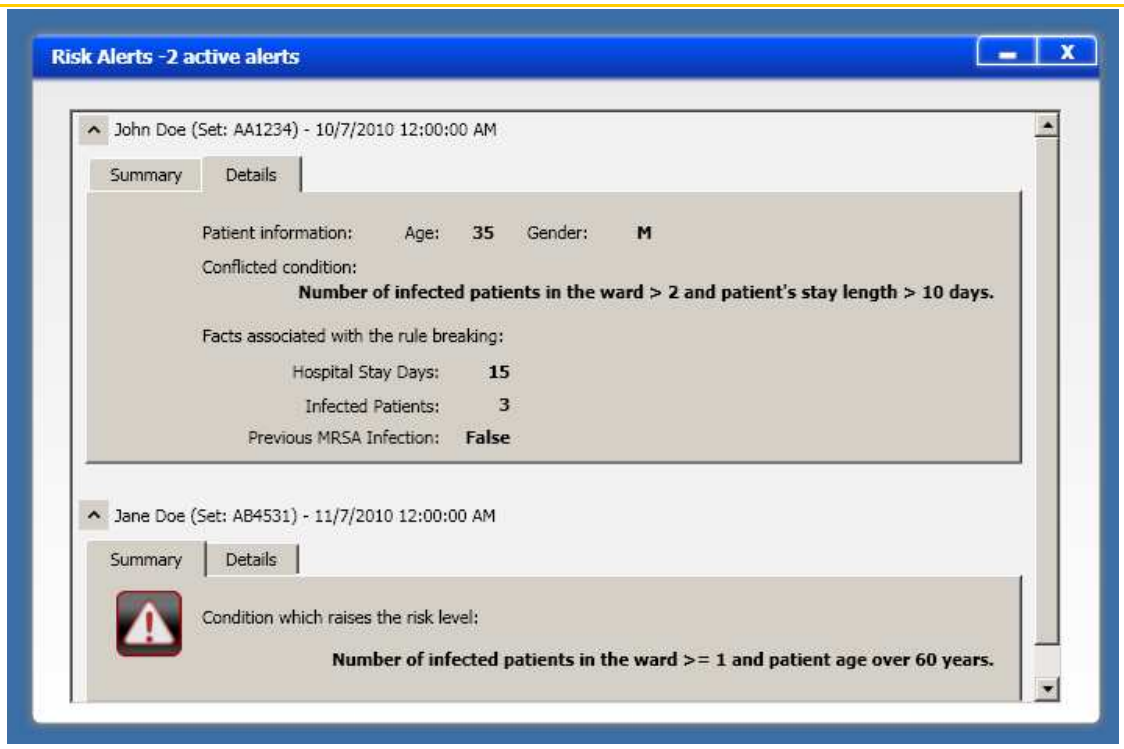

Figure 22. Risk Manager Interface. Details of first alert

Figure 23. Risk Manager Interface. Details of second alert

6. Conclusions

Deliverable D3.8 – Third revision of the data mining and knowledge extraction module addresses the implementation of data mining component and the modelling of the selected mining scenarios, as also the development of the Knowledge Inference System and of the Risk Manager Interface, which consists in the risk manager interface for the alerts produced by the Knowledge Inference System.

The present deliverable documents the development of the TRFT data mining scenarios, which aims at providing expressive modelling of the historical data of patient administration and related MRSA infection control activities. The laboratory results which confirms MRSA infections together with patient hospital and ward admission are employed for identifying patterns that help predicting future risky situations for the patients. The resulting mining models were used as the rule base for the Knowledge Inference System.

The Knowledge Inference System was implemented by relying on the well-defined foundation of business rule engines. The deliverable specifies how the mining models are reused in this knowledge-based inference engine for producing reliable risk pattern detection in new, real-time incoming data. The Risk Manager Interface was also developed to deliver easy to use visual alerting capabilities to the risk manager user.

7. Glossary

ADT – Admission, Discharge, Transfer

API – Application Programming Interface

CDA – Clinical Document Architecture

DM&KE – Data Mining & Knowledge Extraction

HL7 – Health Level Seven

KIS – Knowledge Inference System

MSCUI – Microsoft Health Common User Interface

ORU – Laboratory messages for TRFT Pilot

RAPS – Risks Against Patient Safety

RMI – Risk Manager Interface

TRFT – Rotherham Foundation Trust

8. References

[1] CDA implementation guide – included in Deliverable D3.5 as Annex 6: CDA_implementation_guide.doc with CDA_Vocabularies.doc.

9. Annexes

[1] ReMINE TRFT Pilot Application Ontology - D3.8_Annex1_TRTF_ontology.owl