REMINE

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

FP7 Contract: 216134

Due date of deliverable: 30 April 2011
Actual submission date: 31 May 2011
Start date of project: 1 January 2008
Duration: 36 months

Organisation name of lead contractor for this deliverable: INDRA
Revision: V.1.0

PROPRIETARY RIGHTS STATEMENT
This document contains information, which is proprietary to the REMINE Consortium. Neither this document nor the information contained herein shall be used, duplicated or communicated by any means to any third party, in whole or in parts, except with prior written consent of the REMINE Consortium.
# Table of Contents

1 INTRODUCTION .............................................................................................................. 5  
1.1 PROJECT OVERVIEW .................................................................................................. 5  
1.2 CONTEXT OF THE RELATED TASK WITHIN THE PROJECT ...................................... 5  
1.3 CONTEXT OF THIS DELIVERABLE WITHIN THIS TASK/PROJECT ............................ 6  
2 3RD PROTOTYPE SPECIFICATION ................................................................................. 7  
2.1 PROTOTYPE METHODOLOGY ...................................................................................... 7  
2.2 PROTOTYPE COMPONENTS ........................................................................................ 8  
3 3RD PROTOTYPE REPORT ............................................................................................... 12  
3.1 DATA EVENT MANAGEMENT SYSTEM -DEMS .......................................................... 12  
3.1.1 Architecture ............................................................................................................. 12  
3.1.2 Functionality and features ....................................................................................... 14  
3.1.3 Dependencies ......................................................................................................... 19  
3.1.4 Installation, configuration and updating .................................................................... 20  
3.1.5 Technical Challenges ................................................................................................ 21  
3.2 METADATABASE .......................................................................................................... 22  
3.2.1 Architecture ............................................................................................................. 22  
3.2.2 Software development methodology ...................................................................... 23  
3.2.3 Development tools .................................................................................................. 26  
3.2.3.1 Microsoft Visual Studio ....................................................................................... 26  
3.2.3.2 SQL Server Management Studio .......................................................................... 27  
3.2.4 RLUS ....................................................................................................................... 29  
3.2.5 TOS ......................................................................................................................... 32  
3.2.6 Auditing services .................................................................................................... 34  
3.2.7 RLUS WS-Eventing .................................................................................................. 35  
3.2.8 Installation and configuration ................................................................................... 35  
3.3 H-ERP BRIDGE ........................................................................................................... 41  
3.3.1 Introduction ............................................................................................................. 41  
3.3.2 Development .......................................................................................................... 43  
3.3.2.1 (IT) Niguarda ....................................................................................................... 43  
3.3.2.2 (IT) Sacco ............................................................................................................ 43  
3.3.2.3 (UK) TRFT .......................................................................................................... 44  
3.3.2.4 (FIN) SEK ........................................................................................................... 44  
3.3.3 SW modules exported in the ReMINE DVD .............................................................. 45  
3.4 INTALIO PLATFORM .................................................................................................... 46  
3.4.1 Overview ................................................................................................................ 46  
3.4.2 Architecture .......................................................................................................... 46  
3.4.3 Software Development Methodology ................................................................. 48  
3.4.4 Functionality and Features ..................................................................................... 50  
3.4.5 Business Rules (BRs) ............................................................................................ 50  
3.4.6 Security Layer ........................................................................................................ 52  
3.4.7 Graphical User Interface (GUI) ............................................................................... 53  
3.4.8 Installation and configuration ................................................................................. 54  
3.5 DATA MINING AND KNOWLEDGE EXTRACTION ...................................................... 55  
3.5.1 Architecture .......................................................................................................... 55  
3.5.2 Functionality and features ..................................................................................... 56  
3.5.3 Dependencies ......................................................................................................... 57  
3.5.4 Installation, configuration and updating ............................................................... 57  
3.5.5 Next steps .............................................................................................................. 60  
3.6 ALERT MANAGING SYSTEM (AMS) .......................................................................... 60  
3.6.1 Architecture .......................................................................................................... 60  
3.6.2 Functionality and features ..................................................................................... 63  
3.6.3 Dependencies ......................................................................................................... 65  
3.6.4 Installation, configuration and updating ............................................................... 65
1 INTRODUCTION

The current document titled “D6.4 – Third Platform REMINE Prototype” is the third of three deliverables. This deliverable presents the development effort for the REMINE platform. In the following chapters, a description will be given about the concept of prototype in the REMINE project, the architecture and the associated functionality of each sub-module implemented for this third development period.

Deliverable D6.4 (3rd iteration) apart from the current report is accompanied with a DVD containing the source-binary code and necessary libraries of implemented modules.

1.1 Project overview

The REMINE framework/system will contribute to the better management and mitigation of Risks Against Patient Safety (RAPS) in local health care systems through the definition of a framework architecture, instantiated and validated in a proof-of-concept prototype.

REMINE will enable the collection and analysis of RAPS-related data, through a semantic approach allowing a fast and secure extraction of data and correlation of the information across several domains. In this respect, the REMINE platform will promote an early RAPS detection and mitigation by supporting the process of RAPS management in both of the following situations:

- When a RAPS is foreseen and the objective is the determination of the optimum set of preventive actions, the method of communicating these to the key actors (of the patient care process) involved, etc;
- When a RAPS is detected (i.e. it has been instantiated) and the objective is the determination of the best possible reaction, the secure and reliable distribution of the related action-list to all the involved parties and the monitoring of the reaction effectiveness.

These important capabilities will be achieved by means of the establishment of an associated methodology and a framework/platform for integrated RAPS prediction/detection, analysis and mitigation. The overall platform structure assumes the presence of an “info broker patient safety framework” connected with the Hospital Information System (in following referred to as “HIS” or “ERP” or “H-ERP”) which will support the process of collection, aggregation, mining and assessment of related data and distribute advise/alerts and/or suggest actions to mitigate (or avoid) RAPS effects (or even occurrence). The underlying ontological system will support the semantic correlation of data with the hospital processes.

1.2 Context of the related task within the project

Deliverable D6.4 corresponds to task T6.2 titled “Component Development Integration and QoS and Test Plan” which has two main objectives, the development of several modules that are going to comprise the REMINE platform and the definition of deployment environment, integration methodology and test
plan. This deliverable is part of the first objective and the work associated in designing, implementing and testing the REMINE 3\textsuperscript{rd} Prototype.

1.3 Context of this deliverable within this task/project

The deliverable D6.4 as mentioned is the third in the series that will specify the REMINE platform in terms of components, architecture, implemented functionality and features as well as listing technical challenges encountered and future development plans.

For the delivery of this report and the accompanied DVD disk that contains the source(binary code, several other deliverables have been studied and in particular D6.1 REMINE Architecture Specification; D6.2 Software Component Test Plan and QoS Evaluation Metrics and D6.3 First Platform REMINE Prototype.
2 3rd PROTOTYPE SPECIFICATION

The role of a prototype in a software project has many benefits and it is an essential incremental step towards the implementation that will result to the final software product ready for the production environment. In REMINE project consist of three prototypes, which aim to test the technologies, architecture, component features and integration aspects.

The third prototype aims to be the final solution for the four different scenarios. This third prototype is an enhancement to previous prototypes, being the platform to be used as a validation for each hospital.

![Third prototype architecture overview](image)

Figure 1: Third prototype architecture overview

2.1 Prototype methodology

In order to simplify and manage the development lifecycle of REMINE software prototype, a custom methodology titled Dynamic Systems Development Method (DSDM) has been followed. Although this methodology defines a wide range of activities across various domains, from design to deployment, only a specific subset has been used in the context of REMINE project. This loose adaptation of the aforementioned methodology is in-line with modern best practices for software development such as agile methods where customers are encouraged to select activities that only create value in their projects rather than following one strict methodology plan.

For REMINE project, the top-level lifecycle of DSDM has been used where iterative and evolutionary steps assist in the continuous refinement of the software and management practises. This lifecycle consists of:

1. Identification of the prototype (scope, goals, milestones, etc.)
2. Agreement to a development plan between collaborating partners (technical and business)
3. Prototype development and implementation
4. Prototype review from developers and end-users

Figure 2: REMINE prototype methodology
The comprising components of REMINE include a number of modules that work together in order to deliver the associated functionalities to the hospital side. The 3rd prototype consist of:

1. H-ERP bridge
2. Data Acquisition Layer and Data Event Management System (DAL &DEMS)
3. Metadatabase
4. Process Mapper (part of INTALIO platform)
5. User interfaces (part of INTALIO platform)
6. Data Mining and Knowledge extraction
7. Alert Managing System (AMS)
8. Security Layer (part of INTALIO platform)

The following section describes a short overview of the components’ description and aspects present on this third prototype. For a detailed explanation of features for each component please refer to chapter 3 and the corresponding section.

**H-ERP bridge:** This component is a core software module in the REMINE architecture and serves as the “bridge” of the REMINE platform to the hospital system. Its purpose is to create the necessary adapters that sit on top of the pilot IT system and which propagate data to the rest of REMINE system. It shall input data directly from the Hospital Information System (HIS), including such information as patient historical data, test results, images, etc, through a number of dedicated/customised interfaces. This module will also be responsible for the appropriate data collection, selection, ‘cleansing’ and preparation, and also for the communication with ‘Data Acquisition Layer’ module.

**Data Acquisition Layer and Data Event Management System:** This module will be the ‘gateway’ via which the required data will enter the REMINE system. Specifically, this module will receive the data selected (‘pushed’ or ‘pulled’) by the ‘H-ERP Bridge’ component in an HL7/XML format and appropriately ‘normalise’ it with respect to the selected/applicable ‘coding’ schemas (e.g. ICD9/10, SNOMED or other). Afterwards this module’s purpose will be to ‘enrich’/‘normalise’ the data, by ‘associating’ it with specific ‘data events’ (i.e. recognized ‘data instances’ e.g. a pregnant woman’s admission to the Obstetrics Ward) and tagging it in an ‘intelligent’ (possibly semi-automatic) fashion with respect to the REMINE ontology, and in particular with respect to SHEL (Software Hardware Environment Liveware) and Time parameters.

**Metadatabase:** This module forms the main persistence layer (database) of REMINE architecture and provides an extensive API for interfacing. In this first prototype, a fully functional database system is provided with a connectivity layer that provides necessary interfaces to other components. This module comprises of:

- The ‘Metadatabase Reasoner’: a rule engine to check the semantic integrity of the incoming data before it is stored in the REMINE Database.
The ‘REMINE Taxonomy’: will provide a comprehensive mapping of the healthcare system that will enable the data examined by the system (e.g. incident records etc.) to be partitioned into ‘logical’ categories to be used by the REMINE system for its ‘risk mitigation’ services.

The ‘REMINE Ontology’: primarily intended to be used as a detailed domain description elucidating relations between data events, identifying patient risks, considering previous adverse events documented in the system.

The ‘REMINE Metadatabase’: The REMINE ‘Metadatabase’ component will manage all the data activities for the REMINE project. It will contain data that are directly acquired and produced inside the REMINE boundary and data coming from other ‘external’ data sources.

INTALIO processes (Process Mapper): A graphical representation (BPMN) of the hospital processes based on specific scenarios. The different scenarios of Niguarda, Sacco, TRFT and SEK hospitals (pilots) are modelled in BPMN using the INTALIO platform. Then they are deployed (BPEL) to a dedicated server to be compliant with the clinical pathways and provide a flow orchestrating all the medical and informatics services. This module will be responsible for the ‘integration’ of the healthcare organisation’s processes and data via a dedicated application/tool. This module will serve two main functions: the initial mapping of the healthcare organisation’s processes and the association of those with data and ‘Data Events’/‘Adverse Events’.

User Interfaces: Since the executable processes often require user input and a means to present the outcome of a process instance, technologies for user interaction on a web-based environment have been tested. In particular, a proof of concept scenario has been achieved where a user was able to interact with a running process through an HTML form inside a web browser.

The Data Mining & Knowledge Extraction module: This component of the REMINE system will comprise of 2 sub-modules, namely:

- The ’Data Mining’: is dedicated to the extraction of useful ‘patterns’ from heterogeneous clinical and other HIS (and beyond) data with the aid of semantic modelling.
- The ’Knowledge Inference System’: the purpose of this module is to ‘exploit’ the ‘knowledge’ extracted from the data-mining procedures and provide the appropriate ‘answers’ and results for demonstration via the ‘Risky Manager Interface’.

Alert Management System (AMS): is the ReMINE’s component in charge of configure and trace alert events and dispatch correspondent messages according rules executed in the Business Rule Engine. This component was designed to interact with healthcare givers through mail, mobile phones and the ReMINE web-portal (in particular the RMI).
The Security Layer: will be mostly responsible for creating, maintaining and updating records/profiles of the authorised users of the REMINE system, granting secure and safe access to the system’s functionalities for such users, and ensuring that appropriate communication protocols and policies are adopted so as to warrant the safe and uncompromised exchange of information both internally and externally to the system.
3 3rd PROTOTYPE REPORT

3.1 Data Event Management System -DEMS

The Data Event Management System is a core module in the REMINE architecture, which aims to provide the necessary functionality to assist a Risk Manager on his/her daily activities. More specifically, it aims to support the Risk Manager by implementing a rich graphical user interface for specifying electronic “Data Events” in the medical domain, which pose a risk in the hospital environment. This is achieved by utilizing a specialized software tool platform named INTALIO, which is based on the ECLIPSE Rich Client Platform (RCP). In the following sections the first prototype of DEMS module will be presented covering the high-level architecture, functionality and features, dependencies, installation and configuration, technical challenges and finally features to be implemented for the next iterative development period.

3.1.1 Architecture

In order to be able to understand the operation of the DEMS module functionality it is necessary to examine first the REMINE platform architecture and the module’s significant position in the overall system. As seen from the figure below the module resides in the “semantic layer” and interfaces with three components, the ERP-bridge, the database system and the process Mapper. The first is used for feeding the necessary input for both event definition and runtime execution, the second is used for persisting user-designed events and for monitoring of data event triggers and the third is used for driving (invoking) the hospital processes which relate to clinical and non-clinical events.

Figure 4: REMINE platform architecture
The DEMS component is designed in a modular way with clear separation of functionality between sub-modules and thus making it easier to work on new features, maintain code, debug, etc.

The architecture of DAL/DEMS for the first prototype is composed of the following sub-modules:

- Data Event Editor (DEE)
- Ontological Descriptor Tool (ODT) and
- Data Event Runtime Execution Engine (DEREE)
- Data Acquisition Layer (DAL)

The first two sub-modules have a graphical user interface (GUI) while the third is running as a service. Another difference between these two groups of modules is that the first group (DEE, ODT) is being used “on-demand” by the Risk Manager while the latter (DEREE) is an “always-on” service running 24/7. The graphical user interface of the Data Event Management System module is deployed on INTALIO platform and has been implemented with Java SWT graphic libraries of Eclipse RCP. On the other hand, the DEREE sub-component has not front-end and is being developed as a standalone Java application. The DAL sub-module is being developed as a Java web service.

As seen from the above diagram which depicts the inner design of the DEMS component, a n-tier architecture has been adopted. There are 3 distinct layers:
• Presentation layer,
• Persistence layer and
• Business logic layer.

The first is the access-point of the module and provides the required software-user interaction, the second is being used for persisting data event models and ontology annotations and the latter is being used as the runtime execution.

The clear separation of DEMS functionality into layers provides a good design framework to allow future expansion of capabilities, maintainability of code and in general follows modern best practices and software patterns.

3.1.2 Functionality and features

For the first prototype of DEMS module a custom development process that shares methodology with “Unified Process” has been used. The aim was to investigate first what was considered a technical challenge and then continue the development with iterative steps. This approach helps to minimize risk associated with software development as well as provide a complete sub-module with working functionality (albeit not covering 100% of requirements) which can be accessed by other technical partners.

The most technical challenge at the beginning of the development phase for DEMS module was considered the Graphical User Interface and in detail the integration with a large and complicated software platform as the Eclipse RCP. The Eclipse ecosystem provides thousands of library classes, a huge collection of documentation and a specific programming model where programmers should follow diligently. Therefore, an investigation on these resources with hands-on programming exercises would ensure that the steep learning curve is followed in an efficient way, which will provide the optimum results. In addition, the front-end of any application is the component that a customer is more interested on and his/her feedback can be useful for the continuation of the development.

At the beginning of the 1st prototype development period it has been agreed internally and with close cooperation with the technical coordinator to work on the “Data Event Editor” sub-module. The result was a 1st prototype of DEE fully integrated in the INTALIO environment and with a database system and was consisted of several window panes, context menus, dialogs, editors, views, etc. (for more information regarding ECLIPSE RCP terminology please refer http://www.eclipse.org/). The DEE component has been completed (≈ 80-90% complete) although there is always the possibility of adding new functionality as new requirements may arise.
The main features and functionality as per window of DEE is summarized below:

1) Data event view window
   a) Refresh data events
   b) Create data event
   c) Edit data event
   d) Remove data event

2) Data event details window
   a) Add/remove INTALIO processes to a data event
   b) Add/remove conditions to a data event
      i) Add expression
      ii) Add HL7 act
      iii) Add value (numerical )
      iv) Edit operand (Boolean logic, <, >, etc.)
      v) Delete operand

3) Process definition view window
   a) List of all available processes running on the INTALIO server

4) Configuration window
   a) Preferences for the DEMS plug-in
Figure 6: The main view of DEMS with all windows visible

As seen from the above screenshot DEMS 1st prototype is consisted of four windows, which can be re-located, minimized, re-sized, etc. The first window provides a tree-based browsing facility for reviewing existing clinical and non-clinical parameters that can be used to evaluate a condition and trigger a process. The second window is the Data Events Details Window and provides the editor for designing a data event model, assign expressions using the expression editor and finally link them to an existing process. The third window provides a live view of processes that are currently active and running on the INTALIO server. The fourth and last window provides a list view of existing data events designed by the Risk Manager.
Every DEE window (View in Eclipse RCP terminology) can be closed at any time so that other windows can be opened such as the INTALIO Process Designer, Process Explorer, Palette, etc. The Risk Manager can re-enable the view of DEMS windows by clicking Windows > Show View > Other and then selecting “DEMS View Category” option (as shown in the above screenshot).

Although there are four actual windows the fourth titled “Data Event Details” is invoked indirectly through the “Data Event View” window in two occasions. The first is when a “Create a Data Event” option is selected by right clicking and the other when “Edit a Data Event” is performed on an already designed data event. Either action will invoke the Dialog Manager, which in turn will present the “Data Event Details” view in the main window space.
In the "Data Event Details" view the user is able to perform two operations, the first is the assignment of INTALIO processes to a Data Event definition and the other is the definition of an expression which evaluates according to CDA/Non-CDA attributes. As seen from the above screenshot the user has used two parameters, body height and temperature and assigned some values which are evaluated according to Boolean logic.

The CDA/Non-CDA browser provides a multi-level view of available parameters that are presented dynamically by parsing an XML file. The number of parameters and
their children are expected to increase as the project progresses and more hospital services and medical content is made available.

![Configuration editor for DEMS necessary parameters](image)

**Figure 10: The configuration editor for DEMS necessary parameters**

In order for the DEMS module to work properly there are some configuration settings that need to be set the first time the plug-in is installed in the INTALIO platform. As seen from the above screenshot, a dedicated configuration page has been created in the INTALIO preferences dialog under the title "Data View Preferences". This configuration page is divided into 3 separate areas, each providing text fields for defining parameters. The first is being used to locate the XML schemas for CDA/Non-CDA documents which are needed for the browser, the second is used for defining the INTALIO server endpoint in the case the default configuration is not used and the third is used for defining INTALIO API settings necessary to invoke processes in the INTALIO server.

The DAL module is responsible for receiving messages from ERP Bridge, parsing and transforming them into CDA document, and sending the generated CDA document at the storage system.

### 3.1.3 Dependencies

The DEMS component is developed as a module that can be integrated in the INTALIO plug-in architecture (based on the Eclipse ecosystem). Therefore, it cannot be deployed outside the INTALIO application since it makes use of the UI widgets of the hosting platform. Thus, the target platform is INTALIO Designer v6.0.1 and INTALIO Server v6.0.1.

For DEMS persistence functionality, a lightweight Java database (HSQLDB) is used that runs inside Eclipse as a Java project. For the 2nd prototype a transition will be made to a more advanced database (MySQL) which could accommodate also the needs for other components that need to access DEMS persisted data.
The DAL web service is deployed in Apache Tomcat server.

3.1.4 Installation, configuration and updating

DEMS is consisted of a single file packaged as a Java archive that conforms to OSGi bundle specifications.

To install it to the IN Talio platform it is only required that is copied inside the “Plugins” folder under the installed IN Talio folder tree structure.

![Configuration directory tree]

<table>
<thead>
<tr>
<th>Name</th>
<th>Date Modified</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>configuration</td>
<td>29/6/2009 1:42 µ</td>
<td>File Folder</td>
<td></td>
</tr>
<tr>
<td>features</td>
<td>29/6/2009 5:17 µ</td>
<td>File Folder</td>
<td></td>
</tr>
<tr>
<td>jre</td>
<td>29/6/2009 5:18 µ</td>
<td>File Folder</td>
<td></td>
</tr>
<tr>
<td>plugins</td>
<td>29/6/2009 1:42 µ</td>
<td>File Folder</td>
<td></td>
</tr>
<tr>
<td>readme</td>
<td>29/6/2009 5:18 µ</td>
<td>File Folder</td>
<td></td>
</tr>
</tbody>
</table>

Figure 11: Installation folder for DEMS software

After the plug-in has been installed and IN Talio application is run for the first time, the user has to go to Window->Preferences...->Data view preferences and set the CDA/Non-CDA files path as well as the IN Talio server running processes URL (e.g. http://localhost:8080/ode/processes/ProcessManagement).

Since DEMS module uses a database system for persisting and retrieving data events models, the database should be first initiated and running prior to any interaction with the IN Talio platform. For the first prototype of DEE a lightweight Java-based db has been used named HSQLDB. The current implementation of HSQLDB for DEE is deployed as an Eclipse project and it requires an Eclipse runtime environment so that it can be started. The Eclipse project is titled “SimpleDBServer” and the main class is “SimpleDBServer.java”.

Once the main class is invoked, the server will run in an interactive mode accepting commands from the user so that it further tests and optimizations can be performed.

![SimpleDBServer](http://localhost:8080/ode/processes/ProcessManagement)

Figure 12: The HSQLDB database running in interactive mode inside Eclipse
In addition to the database, an Object-Relational Mapping technology has been used by utilizing an open-source framework called Hibernate which has become the standard in ORM technologies. The benefits of using such a framework is the ability to work on high-level abstraction compared to low-level SQL syntax, advanced support error handling and exceptions, transactions support, ability to migrate to a different database system with only minor modifications, etc.

Another important feature of the first prototype of DEMS is the ability to manage its lifecycle in a dynamic way. More importantly, the DEMS plug-in can be updated on-demand by the Risk Manager to a newer version without having to follow a complicate process or requiring customer support assistance. This is achieved by implementing an Eclipse Update Site project which will be deployed in a remote server and all new features will be made available to that repository. When a new version is announced the end-user will be able to update to the new functionality by first setting the remote repository URL (Help > Software Updates > Find and Install... > Search for new features to install > Add Local Site...) and then by selecting the DEM plugin from the “Installed Software” list and clicking the “Update” button.

DAL is consisted of a single file packaged as a Web application archive (RemineDalWs.war). To install it to the Apache Tomcat server it is only required that is copied inside the "webapps" folder under the installed Tomcat folder tree structure.

3.1.5 Technical Challenges

One of the greatest challenges during the 1st prototype period was the steep learning curve of the Eclipse RCP platform and INTALIO API. As with any development framework there’s an initial time required to get acquainted with the new programming models, available tooling, documentation, API and huge library sets and their associated functionality. Once the library packages have been narrowed down and a developing methodology has been established the programming team was more focused on developing the necessary code to fulfill the requirements. Another important challenge which have been encountered was the documentation which is the result of community effort. It has been found that not all times the documentation was accurate and updated to the latest version of both Eclipse and Intalio platforms. Thus, a significant effort has been spent by utilizing the forums and mailing lists to communicate bugs and participate in discussions regarding specific libraries.

The DEREE sub-module is definitely an important aspect of the DEMS module and forms the “heart” and driving force of the component. Since DEREE employs a BRE mechanism for performing the required business logic to evaluate if certain conditions are met, a close research should be performed to the INTALIO BRE component to evaluate its integration with DEMS. The INTALIO BRE is based on a big open-source project called Drools and it is expected that a considerable time will be spent to study this new framework and inner-architecture.

Another important issue to be resolved is whether DEMS should “push” all data pertinent to an executable process to a running instance of an INTALIO process or the running process collect this data from the database directly. A use case exercise
has been agreed between Q&R and INDRA to evaluate advantages/disadvantages of both approaches and the outcome of this will drive the development for this particular issue.

3.2 Metadatabase

3.2.1 Architecture

Figure 12 represents the WP3 place in the data-flow architecture of the whole ReMINE project. As seen, WP3 consists of three different components: the Process Mapper, the Database (database service with reasoner) and Data Mining & Knowledge Inference.

Figure 13: The place of WP3 in ReMINE architecture
The INTALIO Process Mapper in the semantic annotation layer will store processes in the database through the Retrieve Locate and Update Service (RLUS) while exchanging messages with the Data Event Management System. RLUS provides the API for the Database acting like a file system allowing storing many types of resources with appropriate metadata. Metadatabase reasoner (based on Archetype Definition Language) will check the data for semantic integrity before stored in the Database. The Data-Mining process is responsible for applying specific algorithms for knowledge discovery from existing data and for transferring this knowledge to the Database in the form of ‘rules’, intelligent system parameters and trained models.

### 3.2.2 Software development methodology

Microsoft Solutions Framework (MSF) is a set of principles, models, disciplines, concepts, and guidelines for successfully delivering information technology solutions from Microsoft. MSF is not limited to developing applications only; it is also applicable to other IT projects like deployment, networking or infrastructure projects. MSF does not force the developer to use a specific methodology (Waterfall, Agile) but lets them decide what methodology to use. Agile software development refers to a group of software development methodologies that are based on similar principles. Agile methodologies generally promote: a) a project management process that encourages frequent inspection and adaptation; b) a leadership philosophy that encourages teamwork, self-organization and accountability; c) a set of engineering best practices that allow for rapid delivery of high-quality software and d) a business approach that aligns development with customer needs and company goals.

Microsoft Solutions Framework (MSF) for Agile Software Development is a scenario-driven, context-based, agile software development process for building .NET and
other object-oriented applications. MSF for Agile Software Development directly incorporates practices for handling quality of service requirements such as performance and security. It is also context-based and uses a context-driven approach to determine how to operate the project. This approach helps create an adaptive process that overcomes the boundary conditions of most agile software development processes while achieving the objectives set out in the vision of the project. The following section introduces key concepts:

**Roles** - Team members assume one or more roles.

**Workstreams and Activities** - Roles do activities, which are grouped in workstreams. Activities may produce certain work products and may require work products in certain states before they can be performed.

**Work Products** - Work products are the documents, spreadsheets, project plans, source code, and other tangible output from the activities.

**Visual Studio Team System** - Visual Studio Team System provides tools that you can use to enact process guidance.

**Users and Groups** - Team members log in as users. Users belong to groups, which enforce security privileges, and can be used to implement roles.

**Work Item Database and Metric Warehouse** - Everything planned or tracked for the team project is managed in a database. Those records are called work items. They can track the state of activities or tasks within them. Queries and reports let you track your project status health in real time.

**Source Control and Project Portal** - Team Foundation source control stores some of the work products, such as source code and text. The rest, such as documents, spreadsheets, and project plans are stored on the project portal. Spreadsheets and project plans may be linked to the work item database.

Figure 15: Metadatabase development lifecycle
The Software Development Lifecycles includes the following sub processes:

- Planning
- Requirements Analysis
- Design & Initial Development
- Implementation & Coding
- Testing & Integration
- Evaluation
- Release
- Support

In MSF for Agile Software Development, a team of peers is brought together to represent the entire set of constituents that are involved with the production, use, and maintenance of the product. Each team member, or role, is accountable for representing the specific needs of its constituencies and none is more important than another. Together these views provide the necessary checks and balances to ensure that the team produces the right solution.

Roles in MSF for Agile Software Development:

- Business Analyst
- Project Manager
- Architect
- Developer
- Tester
- Release Manager

![Figure 16: Team of Peers](image)

The MSF Agile process template contains the following work item types: A work item is a database record, which Visual Studio Team Foundation uses to track the assignment and state of work. The MSF for Agile Software Development process defines five work items to assign and track work. The common work item database and metrics warehouse make it possible to answer questions on project health in real time.
Scenario - A scenario is a type of work item, recording a single path of user interaction through the system. As the persona attempts to reach a goal, the scenario records the specific steps that they will take in attempting to reach that goal. Some scenarios will record a successful path; others will record an unsuccessful one. When writing scenarios, be specific as there are many possible paths.

Quality of Service Requirement - Quality of service requirements document characteristics of the system such as performance, load, availability, stress, accessibility, serviceability, and maintainability. These requirements usually take the form of constraints on how the system should operate.

Task - A task work item communicates the need to do some work. Each role has its own requirements for a task. For example, a developer uses development tasks to assign work derived from scenarios or quality of service requirements to component owners. The tester uses test tasks to assign the job of writing and running test cases. A task can also be used to signal regressions or to suggest that exploratory testing be performed. Finally, a task can be used generically to assign work within the project. On the work item form, certain fields are used only in cases when a task relates to a particular role.

Bug - A bug is a work item that communicates that a potential problem exists or has existed in the system. The goal of opening a bug is to accurately report bugs in a way that allows the reader to understand the full impact of the problem. The descriptions in the bug report should make it easy to trace through the steps used when the bug was encountered, thus allowing it to be easily reproduced. The test results should clearly show the problem. The clarity and understandability of this description often affects the probability that the bug will be fixed.

Risk - An essential aspect of project management is to identify and manage the inherent risks of a project. A risk is any probable event or condition that can have a potentially negative outcome on the project in the future. A risk work item documents and tracks the technical or organizational risks of a project. When concrete action is required, these risks may translate into tasks to be performed to mitigate the risk. For example, a technical risk can set off an architectural prototyping effort. The team should always regard risk identification in a positive way to ensure contribution of as much information as possible about the risks it faces. The environment should be such that individuals identifying risks can do so without fear of retribution for honest expression of tentative or controversial views. Teams creating a positive risk management environment will be more successful at identifying and addressing risks earlier than those teams operating in a negative risk environment.

3.2.3 Development tools

Using the above described methodology both metadatabase services were developed using the following software development tools:

3.2.3.1 Microsoft Visual Studio
Microsoft Visual Studio is an Integrated Development Environment (IDE) from Microsoft. It can be used to develop console and graphical user interface applications along with Windows Forms applications, web sites, web applications, and web services in both native code together with managed code for all platforms supported by Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework and Microsoft Silverlight.

Visual Studio includes a code editor supporting IntelliSense as well as code refactoring. The integrated debugger works both as a source-level debugger and a machine-level debugger. Other built-in tools include a forms designer for building GUI applications, web designer, class designer, and database schema designer. It allows plug-ins to be added that enhance the functionality at almost every level - including adding support for source control systems (like Subversion and Visual SourceSafe) to adding new toolsets like editors and visual designers for domain-specific languages or toolsets for other aspects of the software development lifecycle (like the Team Foundation Server client: Team Explorer).

![Visual Studio IDE](image)

Figure 17: RUS development screen in Microsoft Visual Studio

### 3.2.3.2 SQL Server Management Studio

SQL Server Management Studio is a tool included with Microsoft SQL Server 2005 and later versions for configuring, managing, and administering all components within Microsoft SQL Server. The tool includes both script editors and graphical tools which work with objects and features of the server. The version used in ReMINE metadata development relates to SQL Server 2008.
The current version of SQL Server, SQL Server 2008, (code-named "Katmai"), was released (RTM) on August 6, 2008 and aims to make data management self-tuning, self-organizing, and self-maintaining with the development of SQL Server Always On technologies, to provide near-zero downtime. SQL Server 2008 will also include support for structured and semi-structured data, including digital media formats for pictures, audio, video and other multimedia data. In current versions, such multimedia data can be stored as BLOBs (binary large objects), but they are generic bit-streams. Intrinsic awareness of multimedia data will allow specialized functions to be performed on them. SQL Server 2008 can be a data storage backend for different varieties of data: XML, email, time/calendar, file, document, spatial, etc as well as perform search, query, analysis, sharing, and synchronization across all data types.

A central feature of SQL Server Management Studio is the Object Explorer, which allows the user to browse, select, and act upon any of the objects within the server. It can be used to visually observe and analyze query plans and optimize the database performance, among others. SQL Server Management Studio can also be used to create a new database, alter any existing database schema by adding or modifying tables and indexes, or analyze performance. It includes the query windows which provide a GUI based interface to write and execute queries.

Other SQL Server 2008 related tools used in the development of the ReMINE metadatabase are:

- Database Engine Tuning Advisor: enables tuning databases for improved query processing. Database Engine Tuning Advisor examines how queries are processed in the databases you specify and then it recommends how you can improve query processing performance by modifying physical design structures such as indexes, indexed views, and partitioning.
- SQL Server Profiler: shows how SQL Server resolves queries internally. This allows administrators to see exactly what Transact-SQL statements or Multi-Dimensional Expressions are submitted to the server and how the server accesses the database or cube to return result sets.
Figure 18: SQL Server Management Studio used in RLUS database development

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

- **Service contracts** describe which operations you can perform on the service.

- **Data contracts** define which data types are passed to and from the service. WCF defines implicit contracts for built-in types such as int and string, but you can easily define explicit opt-in data contracts for custom types.

- **Fault contracts** define which errors are raised by the service, and how the service handles and propagates errors to its clients.

### 3.2.4 RLUS

RLUS (Retrieve Locate Update Service) is a service developed for the 1st prototype that provides the location, retrieval, and update of clinical and non-clinical information. In ReMINE project RLUS represents the API interface layer of the database. It allows the management of documents (clinical and non-clinical) with dynamic structure. RLUS can be configured to accept different metadata information for a specific type of information (documents) through the configuration of slots (metadata information).
A full description of the RLUS service along with WSDL can be found in deliverable D3.5 First Revision of the WP3 framework.

The current status of the RLUS service:

- RLUS has been developed to allow storage of CDA and non-CDA documents within the 1st prototype using Microsoft .Net Framework 3.5 platform and database storage using MS-SQL2008 server.
- Developed Java and .Net client applications to aid technical partners in accessing RLUS web service and for RLUS as a whole testing framework.

RLUS is the main API layer for the metadatabase, the database layer is very important in the development of the metadatabase module:

![RLUS main database tables](Figure 19: RLUS main database tables)

**REGISTRY_ENTRY** – the base abstract table for all RAPS data that will be stored in the metadatabase and contains the main metadata for each RLUS Entry: creation time, description and title of the stored document.

**FOLDER** – direct implementation of an RLUS Entry, it has a modification time column and also a hierarchical reference to allow a tree implementation of folders.

**ACT** – An HL7 conformance base table for RAPS data that extends the REGISTRY_ENTRY by adding new metadata: a hash code to do a CRC check versus, the binary data in the repository, mime type where it is possible to set the type of the stored type, Uri that points to an external repository, a version number and several identification data.

**PARTICIPANT** – can only store information regarding a person (patient, physician, etc) and organizations.

**SLOT** – Tables extending the current metadata;

**SLOT_TYPE_CODE** – contains the available types for new metadata and

**SLOT_VALUE** contains the values for all slots related to a certain ACT. However, mapping the slots with a certain derived type from ACT is done with configuration files in RLUS.

For the testing of RLUS service, two clients were developed, one in .net and one in Java. They can perform RLUS testing on all the exposed web methods described in deliverable D3.5 First revision of the WP3 framework.
Figure 20: RLUS test client in Java
The taxonomy and ontology service (TOS) is based on a HL7 CTS1 (Common Terminology Services v1) implementation. The Health Level Seven Version 3 standards are based on a Reference Information Model (RIM) which is flexible and general in structure. Representation of information within this model is dependent on the availability of terminological resources which can be used to populate the properties of the model with appropriate semantic content. Whenever possible, the HL7 Version 3 standard references existing terminological resources instead of attempting to create a new resource within the standard itself.

The Common Terminology Services (CTS) specification was developed as an alternative to a common data structure. Instead of specifying what an external terminology must look like, HL7 has chosen to identify the common functional characteristics that an external terminology must be able to provide. As an example, an HL7 compliant terminology service will need to be able to determine whether a given concept code is valid within the particular resource. Instead of describing a
table keyed by the resource identifier and concept code, the CTS specification describes an Application Programming Interface (API) call that takes a resource identifier and concept code as input and returns a true/false value.

A full description of the TOS service along with WSDL can be found in deliverable D3.5 First Revision of the WP3 framework.

The current status of the TOS service:
- TOS service has been developed completely using Microsoft .Net Framework 3.5 (based on WCF – Windows communication foundation) with data persistence in MS-SQL 2008 server
- Imported some medical international vocabularies (LOINC, SNOMED-CT, ICD10, HL7 vocabularies). Created vocabulary restrictions (ValueSets) in concordance with the implementation guides for 1st prototype documents.
- Developed Java and .Net client applications to aid technical partners in accessing TOS web service.

The database structure of TOS service is very complex and it is based on LexGrid documentation for HL7 CTS v1 and fully described in deliverable D3.5.

For the testing of TOS service, two clients were developed, one in .net and one in Java. They can perform TOS testing on all the exposed web methods described in deliverable D3.5 First revision of the WP3 framework. It shows current value sets over TOS with concept mapping to pilot concepts using a specific vocabulary language.
Figure 22: TOS test client in Java showing Heart Rhythm Regularity concepts

3.2.6 Auditing services

Audit trail is one the functionalities of the Security Services that was finish for the third revision of the WP3 services, provides the means to address the issues of liability management, asset protection and quality of service. To facilitate a timely response to policy violations, security incidents or infrastructure and application failures, the metadatabase system supports monitoring, logging, analyzing, and reporting on every level of its architecture.

For each audited event, this document specifies the minimal data requirements plus optional data for the following event categories:

1. **Security administrative events** - establishing and maintaining security policy definitions, secured object definitions, role definitions, user definitions, and the relationships among them. In general, these events are specific to the administrative applications.

2. **Audit access events** - reflecting special protections implemented for the audit trail itself.

3. **Security-mediated events** - recording entity identification and authentication, data access, function access, non-repudiation, cryptographic operations, and data import/export for messages and reports. In general, these events are generic to all protected resources, without regard to the application data content.
**Patient care data events** - documenting what was done, by whom, using which resources, from what access points, and to whose medical data. In general, these audits are application-specific since they require knowledge of the application data content.

### 3.2.7 RLUS WS-Eventing

WS-Eventing describes a protocol that allows Web services to subscribe to or accept subscriptions for event notification messages. A mechanism for registering interest is needed because the set of Web services interested in receiving such messages is often unknown in advance or will change over time. This specification defines a protocol for one Web service (called a "subscriber") to register interest (called a "subscription") with another Web service (called an "event source") in receiving messages about events (called "notifications" or "event messages"). The subscriber may manage the subscription by interacting with a Web service (called the "subscription manager") designated by the event source.

To improve robustness, a subscription may be leased by an event source to a subscriber, and the subscription expires over time. The subscription manager provides the ability for the subscriber to renew or cancel the subscription before it expires.

Event sources may deliver events to event sinks by many mechanisms. This specification provides an extensible way for subscribers to identify the delivery mechanism they prefer. While asynchronous, pushed delivery is defined here, the intent is that there should be no limitation or restriction on the delivery mechanisms capable of being supported by this specification.

### 3.2.8 Installation and configuration

i) Prerequisites installation:

RLUS and TOS require Microsoft .Net Framework 3.5 with Service Pack 1 prior the installation (can be downloaded on the Microsoft web site). Microsoft SQL 2008 server is required as the Database Management System. A standard setup is suffice, however the following script must be ran on the master database to enable .net common language runtime on SQL:

```sql
sp_configure 'clr enabled', 1
GO
RECONFIGURE
GO
```

The following directory structure applies to the metadatabase services:

```
+---BD
|   \---Backup
+---CLIENT
|   \---ManagementConsole
|   |   \---EVS.Import.Export.Console
+---DOCUMENTS
|   +---CS&VS
```
ii) Database installation:

The BD folder will contain the SQL server data files once the databases are restored from the BD\Backup files:

- TOS.bak contains the database backup for TOS service
- RLUS.bak contains the database backup for RLUS metadata database
- RLUS_Repository.bak contains the database backup for RLUS Repository database

Using *Microsoft SQL Management Studio* on the pilot server, databases will be restored using the previous files:

![Figure 23: RLUS Database restoration screen 1](image-url)
Having selected the RLUS.bak file in the source for the database the admin can select the file name with complete path where the database data will reside (e.g. D:\WP3_Services\BD\RLUS_data.mdf, D:\WP3_Services\BD\RLUS_log.ldf).

The databases name are: RLUS, RLUS_Repository, TOS. If the admin requires another name for any database, he must update the connection string in the services configuration files (described below).

Figure 24: RLUS Database restoration screen 2

Once all databases have been created (through restoration) the following SQL server database data file will appear (if used the same name for the file – not mandatory):
iii) Services installation:

After the admin will copies the above directory structure with the binaries, further configuration can be done on the services. RLUS and TOS are exposed as web services via HTTP channels and hosted within windows services (each with its own executable):

- For RLUS: ..\RLUS\RLUS.WinService.Host.exe with configuration file RLUS.WinService.Host.exe.config
- For TOS: ..\TOS\WinServiceCTS.exe with configuration file WinServiceCTS.exe.config

In the configuration file (WCF – windows communication foundation specific) the service endpoints can be modified (e.g.: http://localhost:10000/RLUSAdministrative) – used if that specific port is already opened:

For RLUS the following lines in the configuration file can be changed and replace the URI with any other valid URI for web services.

```
<addbaseAddress="http://localhost:10000/RLUSRuntime"/>
<addbaseAddress="http://localhost:10000/RLUSAdministrative"/>
```

For TOS, the URL for services can be modified in the configuration files:

```
<addkey="Xaml.ServiceEndpointURL"
    value="http://localhost:10000/Xaml"/>
<addkey="VocabularyRuntime.ServiceEndpointURL"
    value="http://localhost:10000/CTSvocabulartoyRuntime"/>
<addkey="VocabularyBrowse.ServiceEndpointURL"
    value="http://localhost:10000/CTSVocabularyBrowse"/>
<addkey="MessageRuntime.ServiceEndpointURL"
    value="http://localhost:10000/CTSMessageRuntime"/>
<addkey="MessageBrowse.ServiceEndpointURL"
    value="http://localhost:10000/CTSMessageBrowse"/>
<addkey="MessageEdit.ServiceEndpointURL"
```
If the admin has selected different databases names than the ones presented as defaults, each service has a DBRepository folder where the SQL connection string can be found in the connections.xml file. For RLUS there are 3 connection strings (2 to RLUS and 1 to RLUS_Repository database):

```
<MyConnection>
  <ConnectionID>5</ConnectionID>
  <ConnectionString>data source=localhost;persist security info=False;initial catalog=RLUS;Integrated Security=SSPI;packet size=4096;</ConnectionString>
</MyConnection>
<MyConnection>
  <ConnectionID>6</ConnectionID>
  <ConnectionString>data source=localhost;persist security info=False;initial catalog=RLUS_REPOSITORY;Integrated Security=SSPI;packet size=4096;</ConnectionString>
</MyConnection>
<MyConnection>
  <ConnectionID>9</ConnectionID>
  <ConnectionString>data source=localhost;persist security info=False;initial catalog=RLUS;Integrated Security=SSPI;packet size=4096;</ConnectionString>
</MyConnection>
```

Each of the ConnectionStrings node can be modified to match the database the admin restored. For the TOS service there is only one connection to the TOS database:

```
<MyConnection>
  <ConnectionID>1</ConnectionID>
  <ConnectionString>workstation id=localhost;packet size=4096;data source=localhost;persist security info=False;initial catalog=TOS;Integrated Security=SSPI;Connect Timeout=6000</ConnectionString>
</MyConnection>
```

Next the admin has to create the windows services for RLUS and TOS using the "sc create" windows shell command. For example, having a user ‘user’ with password ‘password’ belonging to a windows domain ‘domain’, the batch looks like:

```
>sc create ReMINE.RLUS binPath= "D:\WP3_Services\RLUS\RLUS.WinForms.Host.exe" start= auto obj= user@domain password= password
```
>sc create ReMINE.TOS binPath= "D:\WP3_Services\TOS\WinServiceCTS.exe " start= auto obj= user@domain password= password

From the “Services” management console the admin can start the services:

Figure 26: Metadatabase installed services

Once the services started, the admin can test the WSDL in any browser based on the URL given to each service:

Figure 27: Accessing RLUS WSDL

In the CLIENT\ManagementConsole\EVS_Import_Export_Console folder there is a setup kit for a management console used to import/export vocabularies in TOS
service. The excel files containing the vocabularies and valueSets required for ReMINE found in the implementation guides (DOCUMENTS\Guides) can be found in DOCUMENTS\CS&VS. The admin does not need to import them in TOS, as they are already imported.

3.3 H-ERP bridge

3.3.1 Introduction

As described in the past deliverables the H-ERP bridge development is based on an integration middleware (ESB) called JCAPS (Java Composite Application Platform Suite – SUN Microsystem now acquired by Oracle). Please refer to D7.3 (Annex 2) where a brief introduction to JCAPS and its main SW modules is reported. JCAPS is installed in each pilots ReMINE Server in its last version JCAPS 6.2.

The main activities made for this prototype can be summarized as following:

- Installation of the JCAPS platform on each Pilot server.
- Creation of an integrated Domain Service where the developed integrations have been deployed.

![Figure 28: JCAPS - Integration Domain Service](image)

- Creation of the specific WebServices for connection between the H-ERP bridge and the other ReMINE SW modules via Data Acquisition Layer.
  - Push WebService exposed by Data Acquisition Layer and used by H-ERPBridge for “pushing” all the HL7 messages. See the accompanying DVD for the related WSDL and XSD files.
  - Pull WebService exposed by the H-ERP Bridge and used by the Data Acquisition Layer for retrieving (“pulling”) the available type of messages. See the accompanying DVD for the related WSDL and XSD files.
• Creation of specific integration adapters (Projects in JCAPS GUI) for managing all the clinical data required by each clinical scenario for each pilot.
  o Getting data from HIS via different format based on the HIS locally to each pilot.
  o Mapping, transformation and sending of all the HL7 messages to the DAL Push Web Service.

Figure 29: JCAPS – Connectivity MAP

• Test of proper communications
3.3.2 Development
For this third prototype the H-ERPBridge includes all the integration adapters required for all the ReMINE Pilots, as described in the following:

In the following some brief summary of the integrations developed for each pilots.

3.3.2.1 (IT) Niguarda

<table>
<thead>
<tr>
<th>Clinical Event</th>
<th>Sender</th>
<th>Type of Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triage</td>
<td>A&amp;E(CBIM-PIESSE)</td>
<td>HL7 - ADT^A04 Register a Patient Message</td>
</tr>
<tr>
<td>Neuro assessment</td>
<td>A&amp;E(CBIM-PIESSE)</td>
<td>HL7 - ADT^A08 Update Patient Informations</td>
</tr>
<tr>
<td>Lab tests Order</td>
<td>A&amp;E(CBIM-PIESSE)</td>
<td>HL7 - OML^O33 Laboratory Order for Multiple Orders Related to a Single Specimen Message</td>
</tr>
<tr>
<td>Lab samples CheckIN</td>
<td>LIS(NL-DNLab)</td>
<td>HL7 - OML^O21 Laboratory Order</td>
</tr>
<tr>
<td>Lab tests Results</td>
<td>LIS(NL-DNLab)</td>
<td>HL7 - OUL_R22 Unsolicited Specimen Oriented Observation Message</td>
</tr>
<tr>
<td>Radiology tests Order (Head CT Scan)</td>
<td>A&amp;E(CBIM-PIESSE)</td>
<td>HL7 - ORM-001 General Order</td>
</tr>
<tr>
<td>ECG Execution</td>
<td>A&amp;E(CBIM-PIESSE)</td>
<td>HL7 - ADT^A08 Update Patient Informations Message</td>
</tr>
<tr>
<td>Fibrinolitic Treatment</td>
<td>A&amp;E(CBIM-PIESSE)</td>
<td>HL7 - ADT^A08 Update Patient Informations Message</td>
</tr>
<tr>
<td>A&amp;E ReAssessment</td>
<td>A&amp;E(CBIM-PIESSE)</td>
<td>HL7 - ADT^A08 Update Patient Informations Message</td>
</tr>
<tr>
<td>A&amp;E Exit from Clinical Pathway</td>
<td>A&amp;E(CBIM-PIESSE)</td>
<td>HL7 - ADT^A08 Update Patient Informations Message</td>
</tr>
<tr>
<td>A&amp;E Discharge</td>
<td>A&amp;E(CBIM-PIESSE)</td>
<td>HL7 - ADT^A03 Discharge/End Visit Message</td>
</tr>
</tbody>
</table>

3.3.2.2 (IT) Sacco

<table>
<thead>
<tr>
<th>Clinical Event</th>
<th>Sender</th>
<th>Type of Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patient Admission</td>
<td>A&amp;E (Prazision-Isolabella)</td>
<td>HL7 - ADT^A04 Register a Patient Message</td>
</tr>
</tbody>
</table>
### 3.3.2.3 (UK) TRFT

<table>
<thead>
<tr>
<th>Clinical Event</th>
<th>Sender</th>
<th>Type of Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patient Admission</td>
<td>PAS(Totalcare)</td>
<td>HL7 - ADT^A01 Admit/Visit notification message</td>
</tr>
<tr>
<td>Patient Update</td>
<td>PAS(Totalcare)</td>
<td>HL7 - ADT^A08 Update Patient Information message</td>
</tr>
<tr>
<td>Lab. Tests Results</td>
<td>LIS(WinPath)</td>
<td>HL7 - ORU^R01 Unsolicited Observation message</td>
</tr>
<tr>
<td>Patient Transfer</td>
<td>PAS(Totalcare)</td>
<td>HL7 - ADT^A02 Transfer Patient Message</td>
</tr>
<tr>
<td>Patient Discharge</td>
<td>PAS(Totalcare)</td>
<td>HL7 - ADT^A03 Discharge/End Visit message</td>
</tr>
<tr>
<td>Patient Registration</td>
<td>PAS(Totalcare)</td>
<td>HL7 - ADT^A28 Transfer Patient Message</td>
</tr>
<tr>
<td>Patient Details Update</td>
<td>PAS(Totalcare)</td>
<td>HL7 - ADT^A31 Discharge/End Visit message</td>
</tr>
<tr>
<td>Patient Merge</td>
<td>PAS(Totalcare)</td>
<td>HL7 - ADT^A34 Transfer Patient Message</td>
</tr>
<tr>
<td>Change Patient ID</td>
<td>PAS(Totalcare)</td>
<td>HL7 - ADT^A47 Discharge/End Visit message</td>
</tr>
<tr>
<td>Transfer Cancel</td>
<td>PAS(Totalcare)</td>
<td>HL7 - ADT^A12 Cancel of a Transfer event</td>
</tr>
<tr>
<td>Discharge Cancel</td>
<td>PAS(Totalcare)</td>
<td>HL7 - ADT^A13 Cancel of a Discharge event</td>
</tr>
</tbody>
</table>

### 3.3.2.4 (FIN) SEK

<table>
<thead>
<tr>
<th>Clinical Event</th>
<th>Sender</th>
<th>Type of Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patients Admitted</td>
<td>HIS(Tieto - Effica)</td>
<td>HL7 - ADT^A08 Update Patient Information Message</td>
</tr>
<tr>
<td>Patient’s Medications</td>
<td>HIS(Tieto - Effica)</td>
<td>HL7 - RGV^O15 Pharmacy/Treatment Give Message</td>
</tr>
<tr>
<td>Patient’s Diagnosis</td>
<td>HIS(Tieto - Effica)</td>
<td>HL7 - BAR^P12 Update Diagnosis/Procedures Message</td>
</tr>
</tbody>
</table>
3.3.3 *SW modules exported in the ReMINE DVD*

The contents of the DVD for the H-ERP Bridge module consists of 4 folders, one for each pilot site, which include a .Zip file with the complete export of the integrations from JCAPS platform for the specific pilot. From these Zip files the whole integration projects can be restored.

In the following a picture with the content of the Zip File related to one of the pilot.

![Figure 31: DVD](image-url)
3.4 INTALIO platform

3.4.1 Overview

The INTALIO platform is the base to develop three different components of the system: the Process Mapper to control and manage the clinical pathway; the Graphical User Interface to interact with the system and the Security Layer for gaining access to the framework. The core of the system is the Process Mapper, hence this section is focused in its description. The interface and the security layer are supporting modules to provide services to the user and to the Process Mapper itself. Therefore they are treated as part of the Process Mapper.

The Process Mapper will be responsible for the integration of the healthcare organisation’s processes and data via a dedicated application/tool. This module will serve two main functions: the initial mapping of the healthcare organisation’s processes and the association of those with data and ‘Data Events’/‘Adverse Events’.

3.4.2 Architecture

The Process Mapper is making use of the different modules/services of the REMINE prototype described in this section:

RLUS (Retrieve, Locate and Update Service): this service allows the communication between the ReMINE metadatabase and the Process Mapper, allowing extracting the clinical documents to be used in the clinical process and also providing a service for storing the clinical information derived from such clinical process.

AMS (Alerting Management System): this service provides the platform for allowing sending alerts to the medical staff due to clinical protocol violations or to any future risk predicted by the Process Mapper.

SAL (Semantic Annotation Layer): through this system the Process Mapper is obtaining the clinical messages and the needed information for triggering the clinical processes. This component is linked to the HIS (Hospital Information System), hence the relevant information is sent to the Process Mapper to start the different clinical pathways.

GUI (Graphical User Interface): by means of this platform, the different users of the system, from the Risk Managers to the different clinicians, are able to interact with the ReMINE system. It is divided in 2 different sections: Medical Process Interface and Risk Manager Interface. The last one is comprised of:

- Alert Management System interface (AMS)
- Business Activity Monitoring Interface (BAM)
- Business Process Management System Console interface (BPMS)
On the other hand, there are different internal components used by the Process Mapper to perform several actions in order to allow the user interaction, the data storage and the decision-making procedure, among others. The main internal components belonging to the Process Mapper module are:

**Business Rules Engine (BRE):** This module is used to define the business logic of the clinical pathway so it is possible to define policies and protocols to be followed by the process flow.

**Database:** The internal database used by the BPEL engine (Process Mapper), either MySql or Derby depending on the scenario, is used to keep track all the tasks and actions of the user. Also it is used afterwards to calculate and show the different Key Performance Indicators (KPIs) of the system.

**Role-based Access Control (RBAC):** this service is used to authenticate a user in the system, thus gaining access, depending on their roles assigned, to access different system resources.

The different modules of the system are exposed as a web services. Therefore the interconnections between the REMINE executable processes (BPEL) and the different systems, either external as AMS or internal as BRE, are carried out through SOAP messages for accessing to the desired web services, conforming a Service Oriented Architecture (SOA). Thus the different REMINE processes for defining the clinical pathways are developed in BPM to control the business flow and the orchestration of the different web services.

BPM offers several key advantages to define the business processes of an organization. BPM makes a business process absolutely transparent, greatly improving visibility and efficiency. Bottlenecks can literally be seen, and removed. It can show where the most delays are occurring, and where is each transaction stuck as it passes from one stage to another. Data about each and every transaction is logged and can be retrieved as and when required. Therefore, it is possible to analyze accurately what happened. Referencing is also easier as embedded searches allow for data to be picked up as required for study.
3.4.3 **Software Development Methodology**

The methodology for the appropriate use of the processes is separated in three steps: 1) the modelling of the processes in BPMN format 2) deployment in the production server as executable (transformation in BPEL) and make them able to trigger the appropriate web services for each hospital. For making them executable they must be able to accept messages and data and 3) Association with Data Events and enrichment with the appropriate KPIs.

The process descriptions of the different scenarios have been modelled in Business Process Modelling Notation (BPMN) format, using the Intalio Designer tool. These processes describe all the actions and the interactions among all the components of the system, defining the workflow of the business processes.

The processes that were modelled in BPMN were indicated and described by the hospital personnel and were represented in Visio diagrams by the ReMINE consortium. The need for the use of Business Process Modelling is to conclude in a dynamic system that will be able to adapt in everyday’s routine by inserting data, messages and associating with Data Events for having as output the identification of risks and alerts for corrective actions. This will help the risk managers and in general all the hospitals for confronting the risks against patient safety. For these scenarios the risks were also identified. The BPMN (Business Process Modelling Notation) will offer to ReMINE a common language for communication issues and will allow the interaction between the hospital risk managers and the IT users.

There are four different scenarios with their associated processes:

At Niguarda Hospital the scenario is mainly focused on the management of the stroke acute phase that is done in the A&E: it considers the risks caused by a late stroke diagnosis and lack of early information assessment.

At Sacco Hospital the scenario focuses on the main activities of the labour assistance as planned by the clinical protocol: it considers the risk for patient safety due to a delayed execution of the FHR (Fetal Heart Rate) monitoring and obstetrician assessment, because these tasks provide healthcare professionals with important information for the correct assessment of the woman and for the selection of the best clinical pathway.
The scenario of Rotherham Hospital focuses on the prevention and management of hospital acquired infections. It shows how patient history can be analyzed and used by REMINE to prevent the spread of infections and to manage them in case they occur.

At Kauhajoki Hospital the scenario considers the risks related to the process of drug preparation and administration in the Acute Care for the Elderly Unit.

Besides the tasks for each patient, the framework offers a bunch of different processes to control the contextual information of the obstetrics service. In other words, the system tracks the available medical resources and the kind of activity they are carrying out at a given time.

Once developed the different scenarios under BPM language, they will be compiled to BPEL and deployed in a dedicated tomcat server, in order to make them available for the data and events coming from HIS which will trigger and interact with the different processes.
3.4.4 Functionality and Features

The different functions and characteristics of this module are listed:

- Drives the process flow, controlling the clinical pathway
- Creates a framework for the medical service
- Management and administration of clinical resources
- Digital storage of clinical information (CDA stored in metadatabase)
- Detects and alerts of medical protocol violations
- Detects and alerts of future risky situations
- Stores and calculates the Key Performance Indicators
- Interacts with the users through AJAX forms to carry out the communication between the REMINE platform and the medical staff
- Provides security: the users have credentials and roles assigned to gain access to the different parts of the system.

3.4.5 Business Rules (BRs)

This sub-model provides recommendations or solutions; drives the flow of a particular situation in order to control and influence the behaviour of the business within the healthcare service. The business rules aims to establish the guidelines, defining policies and medical protocols, requirements and conditional statements.

ReMINE is a rule-based application, which allow to the clinical responsible of the scenario to drive the scenario and the clinical pathway based on the medical protocols.
Figure 35: Architecture of a rule-based application system.

The structure of the business rule system consists of two inputs (facts and rules) that are applied to the business rules engine in order to produce the results required. The three parts of the BR system are:

- **Facts**: medical information
- **Rules**: guidelines based on clinical protocols, defining:
  - Clinical pathway
  - Protocol constraints
  - Time constraints
- **Results**:
  - Process flow, i.e. clinical pathway
  - Messages and alerts when violation of protocol occurs
  - Messages and alerts when violation of protocol have been foreseen by the system
  - Current and future patient risks and predictions

The BR’s can be divided into two groups:

- **Real Time Business Rules** support real-time management of contingent risky situations for a specific patient, thus their effect on patient safety is immediate.

- **Predictive Business Rules** detect potential future risky situations for a (group of) patient(s), thus their effect on patient safety is in the future.
3.4.6 Security Layer

Due to the environment within the project is developed and the information managed, the security implemented for the data handled by the system and the communications is a critical task. The data collected in the hospital for each patient should be confidential and private, hence the security to protect a potential modification or illegal access to the information or any violation of the patient privacy was one of the main goals of the project. Also the communications among all the subsystems and the different access points to the scenarios have to be secured, as they can present several security problems. As the architecture of the system is service-oriented (SOA) the security over the communications presents an important issue.

The REMINE pilot server is placed within the confines of the local HIS Firewall. The local HIS Firewall and the use of specific user access list (with the associated credentials) are the main security measures for avoiding any malicious access to the platform. Also the access to the different parts is restricted depending on the user roles.

The security is implemented under a role based interface and managed by means of LDAP, taking in account the possibilities of Intalio server security. In other words, the different users of the system need to authenticate and obtain rights and permissions for the different resources and task. It is controlled by the security system, denying unauthorized access depending on the roles of the users assigned by the manager of the system.

The ReMINE login makes use of RBAC. Role-based access control (RBAC) is an approach to restricting system access to authorized users. The permissions to perform certain operations are assigned to specific roles. The users are assigned particular roles, and through those role assignments acquire the permissions to perform particular system functions and components.

The users of the system will have their own credentials to log in the interface with one or more roles assigned in order to allow accessing to the resources with an according level of permission. Every resource, which is out of the scope of the user role, will be hidden to him, offering different interfaces depending on the users. Then every resource has his own role/s associated for controlling the access from the users.

To gain access to Sacco web-portal the user has to authenticate in order to be able to interact with the platform.
3.4.7 Graphical User Interface (GUI)

The framework looks different for each user depending on their roles. The interface automatically enables and disables different components and features to ensure proper use of permissions for the different kind of users. Although there is a unique
workspace, the GUI can be divided in two broad parts: Medical Process Interface (MPI) and Risk Manager Interface (RMI)

The Medical Process Interface (MPI) is intended for the clinical staff and it differs from one scenario to another. Mainly the MPI use to be comprised of a tab for the patient admission; a different one for the different clinical tasks to carry out for each patient; one more for the possibility of launching different medical processes; and finally a tab for notifying the RAPS, showing all the alerts triggered for each patient.

There are special tabs for each pilot as in the case of Sacco scenario that presents a tab to provide the patient risk level and a tab displaying a scheduler with the monitoring activity.

The Risk Manager Interface (RMI) is used for the administrator and manager users. This interface is the same in the four different scenarios. It contains three tabs. The first one, Alert Management System (AMS) tab, for managing all the settings and information related to Alert Management System. The second one, Business Activity Monitoring (BAM) tab, provides the report created in run-time based on the information obtained from the database. The Reports are created according to the Key Performance Indicators (KPIs) defined for each scenario and building be means of different charts. These KPIs provide information about the status of processes; medical information regarding the patients; and detailed statistics of the alerts managed by the system. Finally, BPMS tab, allows the Risk Manager user to check the current situation of the system regarding the execution and the performance of the processes (e.g. lifecycle status, process flow, failures, etc.). Moreover, in case the user has the proper rights and permissions, the console allows performing the following actions: start, stop, deploy and undeploy processes.

---

**Figure 39: Graphical User Interface of Sacco scenario**

---

### 3.4.8 Installation and configuration

In this section an explanation of how to install and configure the different scenarios (attached in the DVD) in the server is given.

1. Before opening the project in INTALIO, some steps are in order to create and use the database:
   - If the scenario use MySql (Sacco)
1. Install MySQL 5.1 if the scenario do not use Derby
   (http://dev.mysql.com/downloads/mysql/5.1.html)
   
   Important note: mysql credentials must be as follows:
   
   user: "root"
   
   password: "admin"

2. Install the connector
   (http://dev.mysql.com/downloads/connector/j/5.1.html)

3. Create the database: execute the sql scripts of the given scenario
   
   • If the scenario use Derby
   
   1. Create the database: execute the sql scripts of the given scenario

II. Unzip the intalio-tomcat server under the folder “DVD_UNIT:\PROCESS MAPPER\INTALIO SERVER”:

III. Customize the server: drag and drop all the folders inside the server tree of the given scenario to the folder of the tomcat server.

IV. Deploy the processes:
   
   1. Open Intalio|Designer and import the projects.
   
   2. Select all and with right click in the project folders select the option “Clean and build now”.
   
   3. Deploy the projects in Intalio|Server.

Now the server is fully configured and installed, in other words ready to use. Open your web browser and type the next address: http://localhost:8080/ui-fw, in order to access to REMINE framework interface. This takes you to the login screen. We should login with an authorized user for the given scenario.

3.5 Data Mining and knowledge extraction

3.5.1 Architecture
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. Given the diversity of the pilots’ scenarios and available data, a different data mining algorithm/technique will be used for each pilot. As a result of this, the implementation technologies of the data mining procedures used for each pilot will be different, but in any case following the same architectural overview. 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 in conjunction with the acquired knowledge from the data mining procedure and send inference alerts to the Risk Manager interface. Namely, current data retrieved from the queue will be evaluated by the KIS under the perspective of the data mining results in order to check for possible alerting situations.

3.5.2 Functionality and features

This module consists of two subcomponents:
  - data mining:
    o acquires historical data from the REMINE Metadatabase
    o applying specific algorithms for knowledge extraction
    o storage of rules definition or other data mining results
  - knowledge inference
o request the rules from data mining
o apply the rules for new incoming data
o suggest of a conclusion or decision
o alert the risk manager

3.5.3 Dependencies

This module depends on:
- the DEMS so that it notifies in real-time all updates in the clinical documents
- RLUS metadatabase – it uses the CDA database to create a minable Data Warehouse
- RISK Manager interface – UI client where it delivers the real time alert by the knowledge extracting module

3.5.4 Installation, configuration and updating

iv) Prerequisites installation:

DMKE requires Microsoft .Net Framework 3.5 with Service Pack 1 prior the installation (can be downloaded on the Microsoft web site). Microsoft SQL 2008 server is required as the Database Management System. A standard setup is suffice, however the following script must be ran on the master database to enable .net common language runtime on SQL:

```
sp_configure 'clr enabled', 1
GO
RECONFIGURE
GO
```

In the existing WP3 services structure, CDAService must be deployed:

```
+----BD
  |   \---Backup
  +----CLIENT
  |      \---ManagementConsole
  |          \---EVS_Import_Export_Console
  +----DOCUMENTS
  |    +----CS&VS
  |    \---GUIDES
\---SERVICES
   +----RLUS
      |   +----DBRepository
      |      \---RegistryEntryDefinitions
      +----TOS
      |    +----DBRepository
      |    \---Temp
\---CDAService
```
v) Database installation:

The BD folder will contain the SQL server data files once the databases are restored from the BD\Backup files:
- HM_Reports.bak contains the database backup for the database required for CDAService

Using **Microsoft SQL Management Studio** on the pilot server, databases will be restored using the previous files:

![Figure 41: CDAService database restoration](image)

vi) Service installation:

After the admin copies the above directory structure with the binaries, further configuration can be done on the services. CDAService has some minor parameters that can be configured.
In the configuration file (WCF – windows communication foundation specific) of the CDAService (CDAService.exe.config) in the appSettings key configuration the following keys can be found:

```
<!--Number of concurent threads used to gather CDAs-->
<add key="WorkerThreads" value="25" />
<!--whether to ignore deleted CDAs-->
<add key="IgnoreDeprecated" value="True" />
<!--timespan when to retry failed CDAs-->
<add key="RetryAfter" value="00:10:11" />
<!--timespan for the start delay -->
<add key="StartDelay" value="00:05:00" />
<!--time of day when to start processing CDAs -->
<add key="StartTime" value="05:00:00" />
<!--time of day when to stop processing CDAs -->
<add key="EndTime" value="22:00:00" />
```

If the admin has selected different databases names than the ones presented as defaults, each service has a DBRepository folder where the SQL connection string can be found in the connections.xml file. For CDAService there are 3 connection strings (1 to HM_REPORTS, 1 to RLUS and 1 to RLUS_Repository database):

```
<?xml version="1.0" encoding="utf-8"?>
<!--Sample XML file generated by XMLSpy v5 rel. 3 U (http://www.xmlspy.com)-->  
<MyConnections xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="D:\MediCross\Server\AdapterRepository\Connections.xml"  

<!-- HM_REPORTS -->
<MyConnection>
  <ConnectionID></ConnectionID>
  <ConnectionString data source=localhost;persist security
Each of the ConnectionStrings node can be modified to match the database the admin restored.
Next the admin has to create the windows services for CDAService using the "sc create" windows shell command. For example, having a user 'user' with password 'password' belonging to a windows domain 'domain', the batch looks like:

```bash
>sc create ReMINE.CDAService binPath="D:\WP3_Services\CDAService\CDAService.exe" start= auto obj= user@domain password= password
```

### 3.5.5 Next steps

Until now the data mining module has been implemented and installed for one pilot (TRFT). The next steps include the implementation of the module for one more pilot (Niguarda hospital). Historical data have already been retrieved and have been anonymized for this pilot. The selection of the proper data mining algorithm regarding the business objectives of this pilot’s scenario, is under development and the future work includes the implementation and installation of the data mining module on the REMINE platform prototype installed on the premises of this pilot.

### 3.6 Alert Managing System (AMS)

#### 3.6.1 Architecture

The identified requirements made clear that there are two main functionalities/flows associated with the alerting system:

- The configuration of the ReMINE alerts (alert types and recipients) should be done on the Risk manager interface, and it should be independent of the processes that might generate them.
- The delivery of real-time (critical, mostly clinical) alerts, according to the level of risk (high, medium, low) through the media identified in the
configuration of the alert and to the recipients defined by the Risk Manager through his administration interface.

These two functional areas are be implemented as two independent modules in order to improve maintainability by enforcing logical boundaries between components. The following diagram depicts the two functional areas as they are implemented and integrated into the Remine solution.

Employing the above approach in REMINE most importantly ensured the following aspects:

- The appropriate ‘actors’ are managing critical alerts by having the appropriate knowledge
- The recipients of the alerts are managed in real time and can be modified at any moment in time.
Separation/distribution of the two modules allows for the distribution of the workload between different servers.

The rules and intelligence to determine the need for an Alert and what type of alert shall be generated were implemented on the Intalio processes, and were adapted with each pilot local needs and rules. The boundaries and interactions of the Alert System are depicted in the next diagram.

The Alert module integrates with an internal system, namely the INTALIO that will be the main source of new alert instances. The only other REMINE actor is the Risk manager that will interact with the AMS through the Risk Manager Interface. The previous diagram also identifies another two actors, named simply as “eMail” and “SMS”. The interactions with these outside systems are properly encapsulated in separated classes, however they represent the two media channels that AMS uses to communicate with the “outside world” and through which the outside world may provide some feedback into AMS.

Finally the next diagram presents the physical architecture adopted in AMS:
They current version of AMS relies in the following machines, and roles:

- **REMINE Server**, currently holds most REMINE functionality including the AMS database and the Risk Manager Web application which includes the AMS managing components.
- **Reverse Proxy – DMZ Server**, which only function is to receive the SMS acknowledgment message from the SMS hosting service and then update the corresponding alert stopping the escalation process.
- **Email Server**, this is the email server available in the local pilot and that is being used to relay the messages generated by AMS to their destination.
- **SMS Hosting Server**, this is a machine that hosts the SMS service and is controlled by the SMS provider service.

### 3.6.2 Functionality and features

Present section enumerates AMS functionality without providing much detail, since this module is fully described on the D2.4, so if any question or doubt remains the reader is encouraged to consult the mentioned deliverable.

Presently AMS functionality may be divided into 2 major groups, the first sets the rules that allows the definition of Alerts and how the system shall process those and to whom shall the information be sent, and the second group are the media channels presently supported by the module. We will describe next the functionalities available in each of these groups.

**Alerts, escalation rules and intended alert targets.**

The following diagram presents, in a simplified way, the several entities managed by the AMS and that support its functionality.
Understanding this data structure will make it simpler to understand the main AMS functionalities, and how to make full use of its potential.

The Risk Manager will be able to manage all AMS information through the AMS Risk Manager Interface, namely he will be able to:

- **Manage Alert Types**, changing its recipients and escalation rules.
- **Manage Users**, or recipients, and user groups.
- **View Alert instances** in a period of time.

The information building block for AMS is the Alert Type, through which the Risk Manager will define the alerts supported by the system. This is also what other modules must define when creating an alert instance on the AMS.

Each Alert type defines 2 sets of information:

- Alert Escalation rules and escalation time.
- Recipients list, which can be a list of specific users, groups or a mix between the two.

The information associated with the Alert Type can be changed at any time by the Risk Manager having its effect immediately applied in the new alert instances and the instances that are still open. AMS client modules should mind this feature at all times, otherwise might expect a different behavior than what is implemented.

**Media channels**

Remine currently supports three media channel through each it registers and sends its Alerts information. Namely:

- **SMS media channel**, where the information is formatted and sent to all recipients mobile phones through an SMS message. This media channel allows also the acknowledgment of the alert through the same media, that is through a formatted reply to the SMS message.
- **Email media channel**, in this case the information is formatted and sent to the recipients by eMail. The acknowledgment of the alert can then be performed by accessing an URL provided in the message itself.
- **Log media channel**, this media channel will not support acknowledgment since its intended use is just to log the Alert information into the database and its not expected to require a user interaction. The information can then be viewed by the Risk Manager through the appropriate Web Interface.
3.6.3 Dependencies

As it is identified in the module architecture, the AMS depends on the following components:

- **INTALIO**, which manages the information workflow and generates new alerts through the AMS web service interface.
- **EMAIL**, namely an email server which AMS interacts through SMTP protocol to send emails to Alert Recipients.
- **SMS**, that identifies an SMS service provider

3.6.4 Installation, configuration and updating

**Required files**

AMS is divided into three major components, and this division is also used in the packages needed to install it, so the present installation instructions expects that the reader has access to the following files:

- Database.sql – MySQL database DDL file, containing the latest database schema.
- AMS.zip – zip file that contains the Web Interface for the AMS management console.
- RemineNotifierWS.aar, archive file that contains the Web Services which exposes AMS functionality to REMINE

**Software pre-requisites**

AMS uses and depends on a SQL database, in REMINE the adopted RDBMS is MySQL, so if none is available it will be necessary to install it before continuing. The other components needed to install AMS are a servlet container, and a Web Service engine. In general AMS needs the following Software Packages:

- Install MySQL version 5.0
- Install Apache Tomcat 6
  - The installation directory (usually "C:\Program Files\Apache Software Foundation\Tomcat 6.0") will be referenced as CATALINA_HOME from now on.
- Install Apache AXIS2 Web Services Engine

**Installation and configuration procedure**

At this point, all the required software should be installed, so the next step is to install and configure the AMS itself:

- Database
  - Import the database file (database.sql) into MySQL server previously installed creating the AMS database instance.
  - Check if the AMS database was properly created.
• Alert Manager Interface
  o Extract the AMS.zip file into the CATALINA_HOME\webapps folder.
  o Edit the properties file (CATALINA_HOME\webapps\RemAlerting\WEB-INF\classes\application.properties) and change its contents as follows:
    ▪ Set database username and password according with what was defined in the MySQL installation and through the existing properties named “db.*”.

• AMS Web Services Services
  o Install the RemineNotifierWS.aar into the services directory of AXIS2 (usually, CATALINA_HOME/wevaoos/axis2/WEB-INF/services)
  o Open the aar archive and edit the application.properties file, updating the following parameters with the appropriate information:
    ▪ Set the Database location and credentials through the db.url, db.user and db.password properties.
    ▪ Set the SMS hosting service URL address along with the necessary credentials to access it through the properties started with “smsservice.*”.
    ▪ Set the SMTP server address that shall be used to send eMails through the properties named “mail.*”
    ▪ Set the log location and file name through the properties named “log.*”
  • Start Apache Tomcat service, if it wasn’t started yet.
  • Check if the interface is up and running by opening the following URL in a web browser: http://localhost:8080/RemAlerting/

3.6.5 Next steps

A few improvements and new features were identified, some of which will depend on the pilot needs and should be analysed more thoroughly. We will name a few that might be useful to guide future effort:
• New delivery alert channels, these might be dependent on existing pilot infra-structure and recipient habits Even existing delivery methods might have usability benefits if adapted to specific equipments. As an example, the SMS media channel might include MultiMedia Messages if the system has the guarantee that the recipients will have equipment that supports this functionalities and there won’t be a security and privacy issues involved.
• Mobile version of Risk Manager Interface to manage the alerts and alert monitoring while on the move.
• Implement even more complex rules to define how to escalate an Alarm.
4 SUMMARY & CONCLUSIONS

The report presented in previous chapters and the accompanied software of the 3rd prototype components forms the deliverable D6.4 that corresponds to task T6.2 and is part of the work of WP6.

The state of the art of the platform presented in this report shows a new prototype where a bunch of new developments to achieve the REMINE goals have been implemented. Among them the integration of all the components, the Alert Managing System, the prediction and alerting of current and future risks, the user of Business Rules (either Real-time BR or Predictive BR), the development of a Risk Manager Interface or KPIs can be mentioned.

Therefore it can be concluded, that REMINE will contribute to the RAPS management process in a local health care system through the developed framework. REMINE third prototype is considered as a proof-of-concept, enabling a collection and analysis of RAPS-related data, through a semantic approach that allows a fast and secure extraction of data and correlation of the information across several domains.
## ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Stands for</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMS</td>
<td>Alerting Management System</td>
</tr>
<tr>
<td>API</td>
<td>Application programming interface</td>
</tr>
<tr>
<td>BPEL</td>
<td>Business Process Executable Language</td>
</tr>
<tr>
<td>BPMN</td>
<td>Business Process Modelling Notation</td>
</tr>
<tr>
<td>BR</td>
<td>Business Rule</td>
</tr>
<tr>
<td>CDA</td>
<td>Clinical document architecture</td>
</tr>
<tr>
<td>CTS</td>
<td>Common Terminology Service</td>
</tr>
<tr>
<td>DSDM</td>
<td>Dynamic Systems Development Method</td>
</tr>
<tr>
<td>ESB</td>
<td>Enterprise Service Bus</td>
</tr>
<tr>
<td>HIS</td>
<td>Hospital Information System</td>
</tr>
<tr>
<td>HL7</td>
<td>Health level 7</td>
</tr>
<tr>
<td>HSQLDB</td>
<td>Hyper Structured Query Language Database</td>
</tr>
<tr>
<td>HTML</td>
<td>HyperText Markup Language</td>
</tr>
<tr>
<td>ICD10</td>
<td>International Classification of Diseases v10</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>LOINC</td>
<td>Logical Observation Identifiers Names and Codes</td>
</tr>
<tr>
<td>MPI</td>
<td>Master Patient Index / Medical Process Interface</td>
</tr>
<tr>
<td>MySQL</td>
<td>My Structured Query Language</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>RAPS</td>
<td>Risk Against Patient Safety</td>
</tr>
<tr>
<td>RCP</td>
<td>Rich Client Platform</td>
</tr>
<tr>
<td>RIM</td>
<td>Reference Information Model</td>
</tr>
<tr>
<td>RLUS</td>
<td>Retrieve Locate and Update service</td>
</tr>
<tr>
<td>RMI</td>
<td>Risk Manager Interface</td>
</tr>
<tr>
<td>SNOMED-CT</td>
<td>Systematized Nomenclature of Medicine - Clinical Terms</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>TOS</td>
<td>Taxonomy and Ontology service</td>
</tr>
<tr>
<td>WCF</td>
<td>Windows Communication Foundation</td>
</tr>
<tr>
<td>WP</td>
<td>Work package</td>
</tr>
<tr>
<td>WSDL</td>
<td>Web Service Definition Language</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>