Deliverable No. 8.6.2

Initial version of the p-medicine integrated platform

Grant Agreement No.:	270089
Deliverable No.:	D.8.6.2
Deliverable Name:	Initial version of the p-medicine integrated platform
Contractual Submission Date:	31/01/2013
Actual Submission Date:	02/04/2013

Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

COVER AND CONTROL PAGE OF DOCUMENT	
Project Acronym:	***p-medicine***
Project Full Name:	From data sharing and integration via VPH models to personalized medicine
Deliverable No.:	D 8.6.2
Document name:	Initial version of the p-medicine integrated platform
Nature (R, P, D, O)[1]	O
Dissemination Level (PU, PP, RE, CO)[2]	PU
Version:	3.0
Actual Submission Date:	02/04/2013
Editor: Institution: E-Mail:	Georgios Zacharioudakis FORTH gzaxar@ics.forth.gr

ABSTRACT:

This deliverable describes the initial version of the integrated p-medicine platform. Based on the p-medicine architecture and selected use cases drawn from the user requirements, an initial version of the integrated platform has been compiled which comprises key software components that demonstrate the vision of p-medicine and pave the way to the complete integrated platform that will comprise all the p-medicine modules that are still under development.

KEYWORD LIST: Integrated platform, software components, architecture, use cases, demonstration scenario

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 270089.

[1] **R**=Report, **P**=Prototype, **D**=Demonstrator, **O**=Other

[2] **PU**=Public, **PP**=Restricted to other programme participants (including the Commission Services), **RE**=Restricted to a group specified by the consortium (including the Commission Services), **CO**=Confidential, only for members of the consortium (including the Commission Services)

MODIFICATION CONTROL			
Version	Date	Status	Author
1.0	01/03/13	Draft	Georgios Zacharioudakis
2.0	15/03/2013	Draft	Georgios Zacharioudakis
3.0	02/04/2013	Final	Georgios Zacharioudakis

List of contributors

- Georgios Zacharioudakis, FORTH
- Stelios Sfakianakis, FORTH
- Manolis Tsiknakis, FORTH
- Haridimos Kondylakis, FORTH
- Lefteris Koumakis, FORTH
- Irini Genitsaridi, FORTH
- Vangelis Kritsotakis, FORTH
- Galatia Iatraki, FORTH
- Alberto Anguita, UPM
- Benjamin Jefferys, UCL
- Ulf Schwarz, IFOMIS
- Elias Neri, Custodix
- Fatima Schera, FHG-IBMT
- Holger Stenzhorn, USAAR
- Juliusz Pukacki, PSNC
- Alberto Anguita, UPM
- Anuj Sharma, Biovista

Contents

1 Executive Summary

The aim of this deliverable is to report on the initial version of the integrated p-medicine platform. Based on the initial p-medicine system architecture and selected use cases drawn from the end user requirements, an initial integrated platform has been composed by the various software components that address selected use cases. This initial version of the platform demonstrates early functionality and creates the core architectural elements, as a first step towards the final integrated p-medicine platform. Also this initial version will expose any early problems or design flaws and will stimulate the discussion for the iterative refinement of the p-medicine architecture. Parts of this integrated platform will be presented during the second annual review of p-medicine project, in an end-to-end demonstrator scenario that combines parts of various use cases.

2 Introduction

2.1 Purpose of this document

The aim of the task 8.6 is "*to deliver an intuitive user level application where users are able to discover the most applicable tools for analysing the data at hand, combine these tools into potentially complex data flows and pipelines, and furthermore to keep an archive of their data processing pipelines, which can be shared and reused by other users*".

To this end, an elaborate user requirement analysis has been performed in D2.2 "*Definition on scenarios and use cases and report on scenario based user needs and requirements*" and characteristic use cases have been documented. Based on these requirements, the initial *p-medicine* architecture has been defined in D3.2 "*Initial System Architecture*" in which it is described how the various software components, will formulate an integrated platform which will provide the functionality mentioned above. In D3.2, a core set of use cases have been selected as the basic required functionality for an initial version of the integrated platform, in order to early expose problems in the architecture, and to provide a minimum set of architectural components in order to gradually create the final integrated p-medicine platform. This is an iterative approach, in which the evaluation from the users and the feedback from the developers will drive the updates to the architecture and the refinement of the platform.

Although all p-medicine components are being implemented and have reached some level of maturity, which may vary to some extent, in this document we present the architectural components that compose the initial integrated platform. That means that we deal mainly with the software components that have been connected with the core modules of the architecture such as the portal or the Data Warehouse and can demonstrate some level of interaction with the rest of the architecture, and not with the components that even though they have reached an acceptable level of functionality, they are still being developed in isolation.

2.2 Structure of the deliverable

The rest of the document is structured as following. In Section 3 we present the p-medicine system architecture and we elaborate on how this affects the integrated platform. In Section 4 we list the use cases that have been selected as exemplar cases to base the initial platform and that helped us select the software components to focus on the initial implementation. In Section 5 we present those software components and their implementation status, their integration with the rest of the platform and we highlight any early remarks or results that that will help refine the system architecture. Finally, in Section 6 we briefly present an end-to-end usage scenario based on the selected use cases that will be demonstrated during the 2nd annual review of the project, to show the current status of the integrated platform and some of its architectural elements.

3 The architecture of the p-medicine platform

In this section we give a short outline of the p-medicine system architecture, which defines the design and drives the implementation of the platform. Based on the architectural design and the set of selected use cases that are realized in priority, the initial integrated platform has been implemented which is presented in Section 4 of this document.

The main requirements and use cases of the p-medicine end users have been described in D2.2 "*Definition on scenarios and use cases and report on scenario based user needs and requirements*". Based on those requirements, an initial version of the p-medicine system architecture has been defined, in D3.2 "*Initial System Architecture*", using the approach of the ISO/IEEE 42010 "*Systems and software engineering – Architecture description*" standard. This initial architecture is subject to iterative refinement, based on the updated user requirements, the feedback from the developers and the evaluation of the platform's prototypes.

Central notions to the architecture are the stakeholders and their requirements, which define accordingly different viewpoints and views of the architecture. The system architecture, based on the different stakeholder groups, defines the following views of the architecture:

- The context view
- The functional view
- The information view
- The concurrency view
- The development view
- The deployment view
- The operational view

As described in D2.2 along with developers, administrators, bioinformaticians, clinical trial managers and other domain experts, the main stakeholder groups in p-medicine are clinicians and patients. To this end, a major part of the definition of the system architecture and subsequently of the p-medicine platform is based on the requirements of clinicians and patients and aims to provide the platform that will address their concerns. Consequently, the initial integrated platform has focused primarily on two views, which mainly address the concerns and requirements of these stakeholders, the functional and the information view of the architecture.

The functional view documents the system's functional elements, their responsibilities, interfaces, and primary interactions. The functional view is the cornerstone of most architecture documents and it drives the shape of the rest of the system structures. It also has a significant impact on the system's quality properties, such as its ability to change, its ability to be secured, and its runtime performance. The information view documents the way that the architecture stores, manipulates, manages, and distributes information. The ultimate purpose of any system is to manipulate information in some form, and this viewpoint develops a broad view of static data structure and information flow.

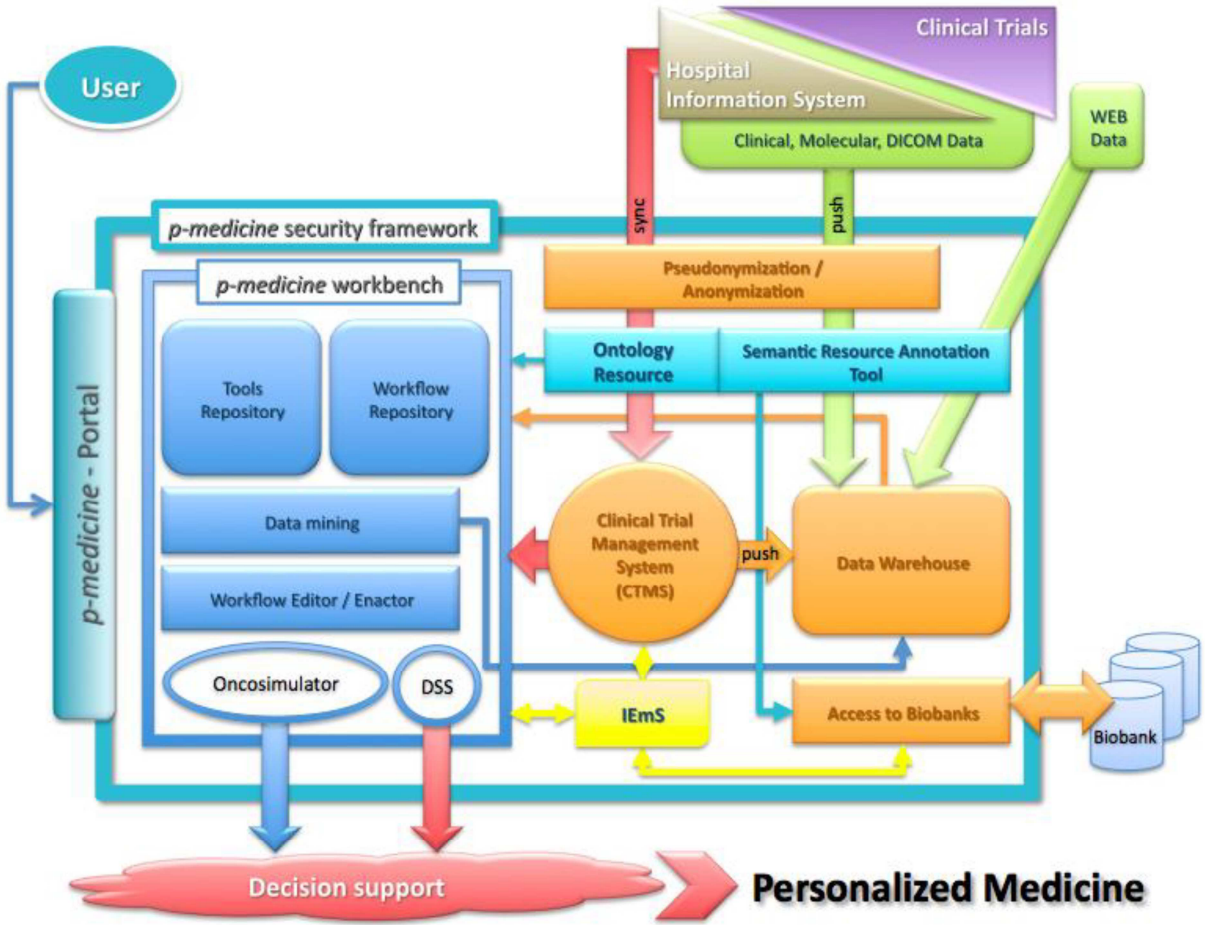

Figure 1 : The architecture of p-medicine from a clinical perspective

The needed functionality is achieved by a large set of software components, either developed internally in p-medicine or provided by third parties and connected to the p-medicine architecture through appropriate APIs (Application Programming Interfaces) or data formats.

These software components can be categorized by two ways. The first categorization is based on the *functional requirements* they address, such as:

- Security tools
- Data Access and Management tools
- VPH Modelling tools
- Clinical Decision Support tools
- Patient Empowerment tools
- …

The second type of categorization is based on the usage scenarios that the components mostly participate, which can be seen as *functional profiles*. This categorization, as depicted in Figure 2 spans across different functional requirement groups and highlights the different types of scenarios that p-medicine targets to address, such as:

- Knowledge Discovery
- Patient Empowerment
- Predictive Modelling
- …

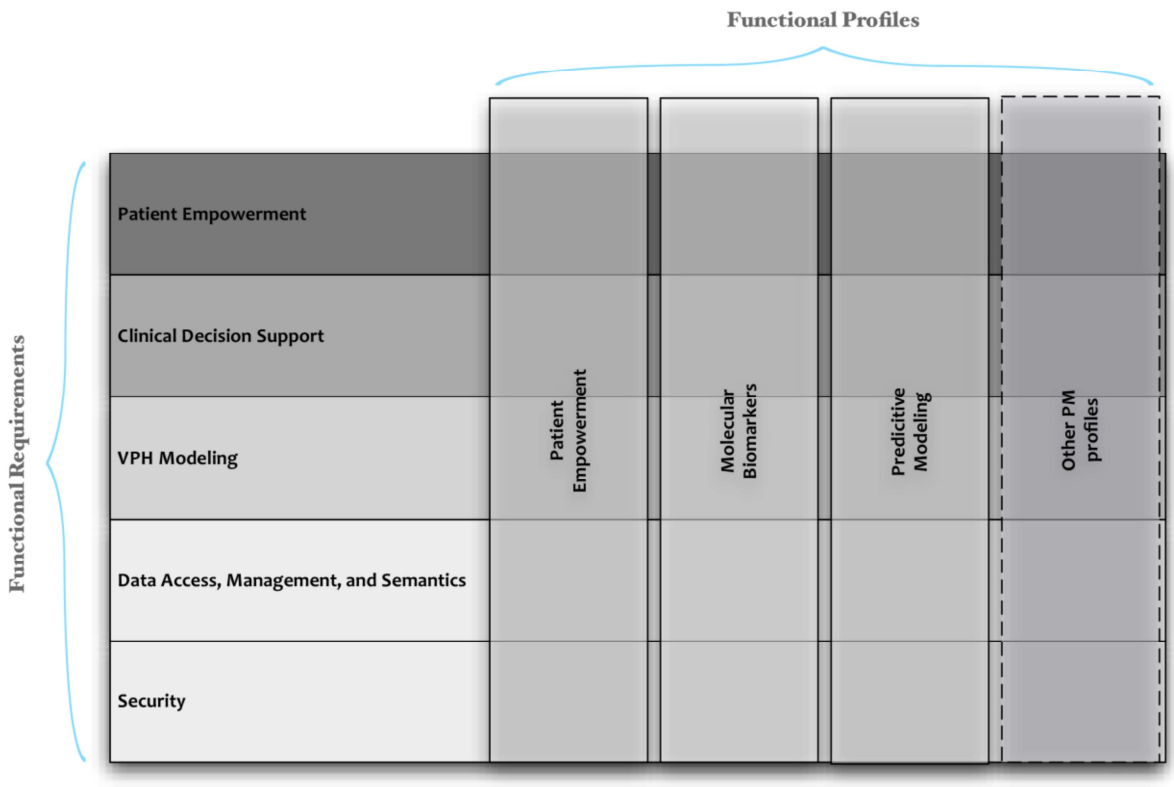

Figure 2 : The Functional (meta)view

In the initial integrated platform, we have tried to include the core components from all different groups and categories, which address both a multitude of functional requirements and participate in different functional profiles so that we can support different usage scenarios.

An elaborate list of the software components being implemented or integrated to the p-medicine platform can be found in deliverable D3.2. In this deliverable we present only a subset which constitutes an initial integrated platform. In Figure 3 are depicted some of the main software components of the system and their interactions, such as:

- The *p-medicine* portal, which is the user interface of the p-medicine platform and the main point of interaction with end users.
- ObTiMA, the clinical trial management system which plays the main role in inserting clinical data into p-medicine.
- The Data Warehouse, the logical repository of data for the p-medicine platform, which relies upon cloud storage services for physically storing the data.
- The Data Mining services, the Clinical Decision Support services (CDSS), the Oncosimulator and the rest of the data analysis services.

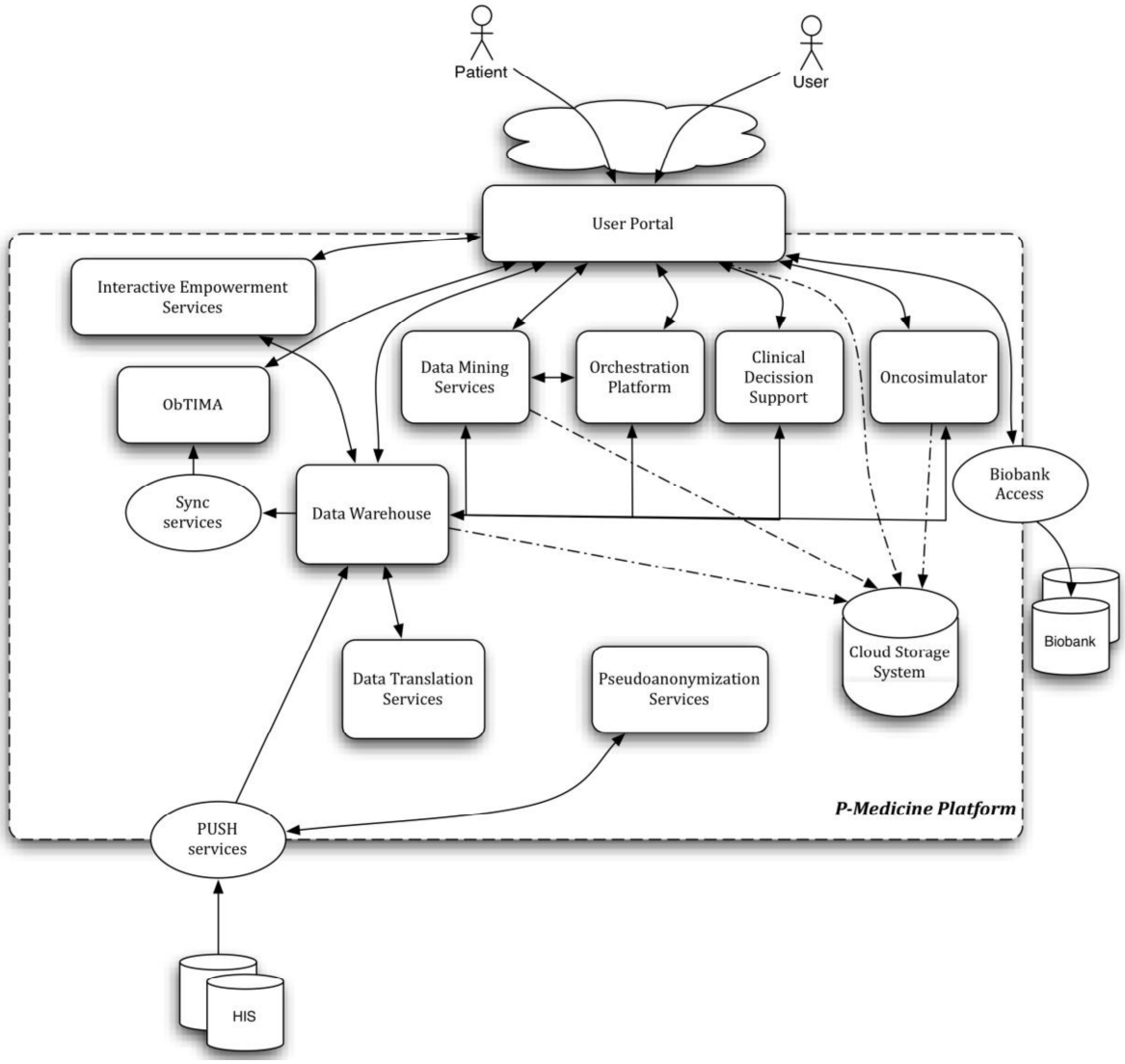

Figure 3 : The main components of the system and their interactions

As it is evident from the usage scenarios and the system architecture, these software components play a key role to the p-medicine ecosystem of interacting components, so the initial integrated platform has focused primarily on them, building a core which can then be expanded to include all components. The experiences and lessons learnt from the integration of this set of components into the initial integrated platform will drive the refinement of the system architecture, as updated versions of D3.2, and the integration guidelines, as updated versions of D8.6.1. Also this set of components provides the initial, development version of the integrated platform, which after testing will be used as a production ready version, for endorsing real clinical data and used in real clinical settings.

4 Selected use cases

In this section we present some selected used cases that were used as the core set of required functionality that the p-medicine platform aims to address in the initial integrated platform, because of their importance and because of their participation in multiple functional profiles, as was described in Section 3.

The selected use cases in D3.2 were categorized as shown in Table 1:

CATEGORY	USE CASES
Generic use cases	Single sign-on and sign-out Authorization Anonymization
Data Flow	Data Translation for Push services User uploading of DICOM images Ontology Annotation of external databases Clinical data analysis of trial data through Obtima Patient Empowerment: Informed consent
Clinically Oriented	Oncosimulator Patient Empowerment: Consent and re-consent Patient Empowerment: Patient enrolment into clinical trials
Knowledge Discovery and Decision Support	Data Mining patterns Clinical Decision Support
Biobank Access	Offering biomaterial to closed or open research community Searching and requesting biomaterial for research Managing biomaterial data in Obtima

Table 1 : Categories of use cases

All these use cases, which are extensively analysed in D2.2, both highlight the core functionality that we focused on implementing in the initial integrated platform, as well as point out the software components that are primarily needed for their realization.

From the generic use cases, the most important were the ones related with the security use cases, since security is a cross cutting functionality that participates in most functional profiles and which is a prerequisite for moving into more complex use cases such as handling of real clinical data. The single sign-on/out and authorization use cases are needed as a basic requirement for a functioning portal, while the anonymization use case is necessary for transitioning from fake or anonymized data into handling data from real clinical trials.

The main software components participating in the Data Flow use cases are shown in Figure 4 and the software components participating in the Clinically Oriented use cases are depicted in Figure 5.

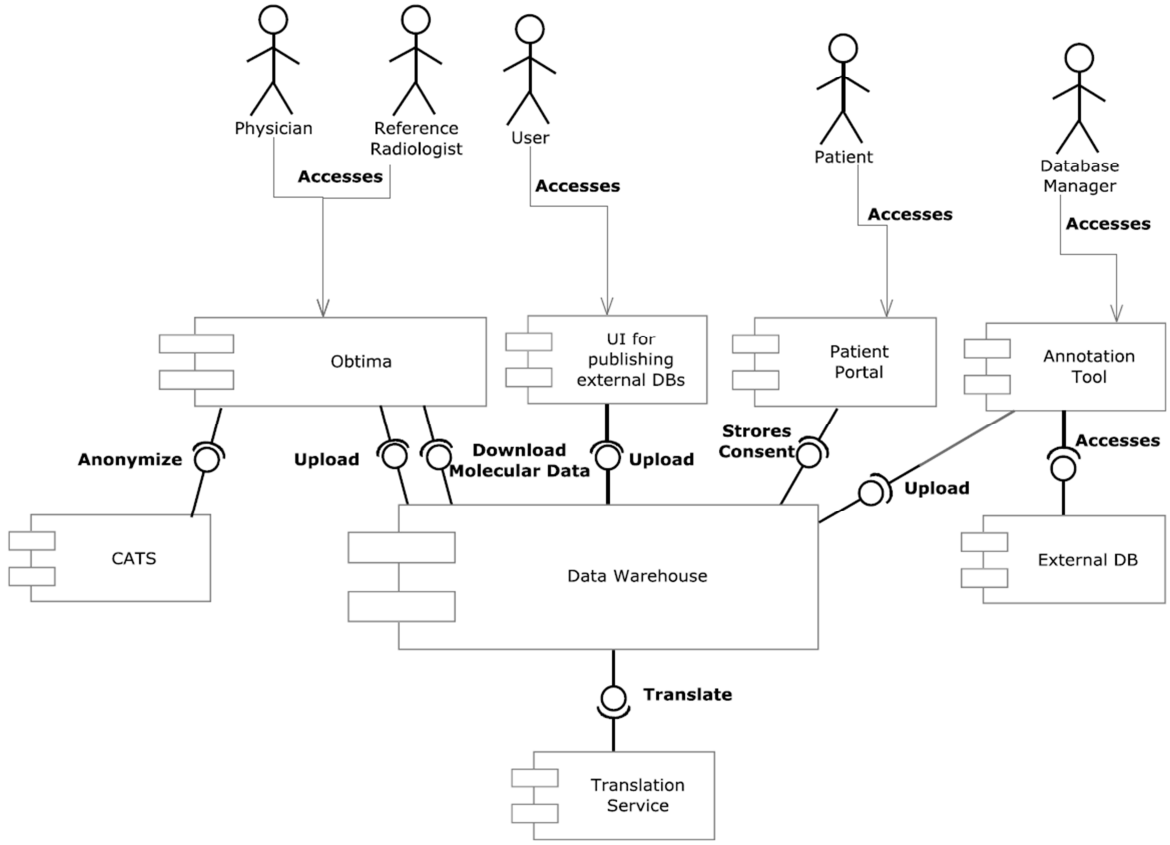

Figure 4 : Component diagram for the Data Flow use cases

From these components, we present in Section 5 their implementation status so far and their integration status with the rest of the architecture.

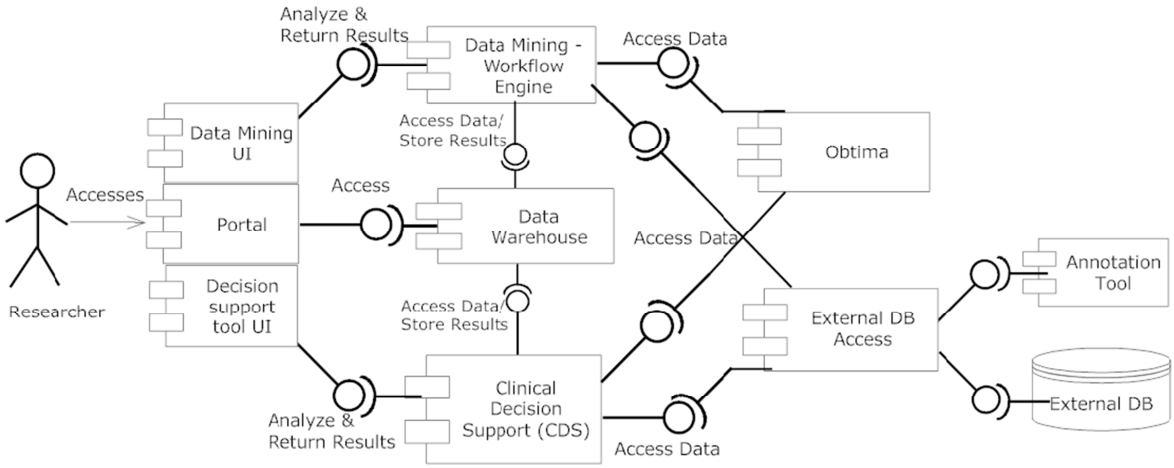

Figure 5 : Component diagram for the Clinically Oriented use cases

5 Software components of the integrated platform

In the previous sections we listed the main software components of the p-medicine architecture, which constitute the core of the platform. Also, these components will be the basis for further expanding the platform by connecting to them other software components that meet the p-medicine requirements and guidelines as they are stated in deliverable D8.6.1 “Integration guidelines and monitoring of tools and services” and its updates.

Below we present the current implementation status of those components and their overall integration with the rest of the platform.

5.1.1 *p-medicine* portal

The *p-medicine* portal is a single web-based environment based on the Liferay framework from which most of the *p-medicine* applications can run. These applications will be integrated together in a consistent and systematic way. The detailed description of the portal is given in the deliverable D8.1.2 “Design and prototype implementation of the p-medicine portal”. Below we present an overview of its current status regarding the integration of p-medicine components into it.

The already included applications in the Liferay framework cover about all of the standard functionality the *p-medicine* users are likely to need in a web site: content management, forums, wikis, blogs, and much more. The collaboration environments provided by Liferay can be used among *p-medicine* user groups: the framework provides applications for document sharing and communicating of portal users. We only need to implement the features specific to the *p-medicine* project.

First of all we have integrated the security framework components (Single Sign-On, Identity Provider for user management in *p-medicine*) which will be used for the whole project. The *p-medicine* portal users will not need to sign in to every application separately. With Single Sign-On functionality the login procedures of different applications will be centralized in one system, accessible with only one password.

At its most basic level the *p-medicine* portal has users, and these users can be grouped together by various ways providing a powerful mechanism for the portal administrator to configure portal resources and security in a consistent and robust manner.

Based on discussions with end users we have defined the following *p-medicine* portal users and their roles in the p-medicine portal, shown in Table 2:

Role in the portal	User groups
CLINICIAN	Clinicians, physicians
CLINICAL TRIAL LEADER	Clinical trial chairman, clinical trial designers
SCIENTIST	Researchers from the scientific community (mathematicians, physicists, bioinformaticians, computer scientists, etc.)
PATIENT	Patients and their relatives
DATA MANAGER	Study nurses, documentalists

ADMINISTRATOR	Technical administrator of the *p-medicine* portal (responsible for installations, configurations, backups, database maintenance, etc.)
PMED ADMINISTRATOR	Administrator of the *p-medicine* portal (responsible for user management, user permissions, community management, etc.)
COMMUNITY ADMINISTRATOR	Administrators of the communities created in the *p-medicine* portal
DEVELOPER (IT)	Developers of tools and services for *p-medicine*
LEGAL AND ETHICAL ISSUES	People dealing with legal and ethical issues
GUEST	Not authenticated portal users

Table 2 : Roles and user groups in p-medicine portal

However, in the future new roles may arise. Different roles giving permissions for using different tools developed under p-*medicine* (e.g. OBTIMA USER) could be created dynamically.

We have decided to use the community based structure of the portal. Portal users belong to Communities that have a common interest (e.g. different portal users are members of a community called **SIOP Community** that has a common interest in the nephroblastoma diseases). Membership in communities gives users access to the pages in the communities of which they're members. Each community can have a specific layout and an own set of available pages containing different *p-medicine* tools. We suggest having only communities with restricted membership in the *p-medicine* portal: users are added to the community by a community administrator.

Communities for specific domains (e.g. ALL, SIOP, Breast Cancer) or specific user collections (patients, scientists, developers) will be created for the *p-medicine* portal dynamically.

Roles described in the table above can appear inside communities. This means that though each community in the portal has this role with its configured permissions, membership in this role is different for each community.

Roles are used to define permissions across their scopes: across the portal or across a community.

The *p-medicine* portal will aggregate the set of portlets representing the *p-medicine* tools that are to appear on any particular page and display them properly to the *p-medicine* user. The portal administrator can arrange the *p-medicine* tools on the portal's pages in the way that works best for the portal users.

The *p-medicine* specific tools and services which should be integrated into the portal are divided into the following two groups:

- *p-medicine* tools that can be integrated **directly** into the portal as plugins:
  - *p-medicine* Workbench
  - ObTiMA
  - p-BioSPRE
  - User data management in cloud services and OpenStack Object Storage
  - Oncosimulator
  - Data Mining Tools
  - Data Warehouse Access Tools

  - Ontology Aggregator
  - Ontology Annotator
  - Patient Empowerment Tools

- *p-medicine* tools that are used as parts of other tools accessed from the portal for performing user workflows which are therefore **indirectly** integrated into the portal:
  - Data Translation Service
  - Sync Services
  - p-BioBank Wrappers
  - ObTiMA Trial Biomaterial Manager
  - dcm4che[3] DICOM server
  - DrEye
  - Data Warehouse Programmatic Interface
  - Workflows and Pipelines for Genomics Data
  - Decision Support Tools

The integration of tools and services developed for *p-medicine* can be performed on different ways:

- A tool for *p-medicine* can be provided as a portlet which is a web application that runs in a portion of a portal's web page. This way is preferred for the *p-medicine* tools which should be developed for the project (e.g. Oncosimulator; Data Warehouse).
- For tools that offer a programmatic interface (e.g. based on Web services, REST, etc.) and have been semantically annotated and registered with the tools repository, the *p-medicine* workbench portlet can be used to present a tool specific user interface, invoke them, and retrieve their results.
- For already existing *p-medicine* tools available as web applications (e.g. ObTiMA) several integration possibilities exist:
  - To re-write the application as a portlet, if it is rational.
  - To create a middleware portlet for interacting with the application (better using web service).
  - To wrap the application as an OpenSocial gadget for displaying it in an iframe. *iframe* is a portlet which makes it possible to embed another HTML page inside the current page.
  - To create a portlet that integrates the application either using an iframe or an HTTP proxy (e.g. using Liferay's WebProxy portlet).
  - If the existing application is a JavaEE application, a Liferay's technology called Web Application Integrator that allows prototyping the integration by deploying the WAR file of the web application can be used. As a result the portal framework will automatically create a portlet that integrates the application using an iframe.

The role assigned to a *p-medicine* portlet developer allows him to deploy the developed portlet into the portal.

There are currently some plugins already available in the portal:

- *p-medicine* theme-plugin for the *p-medicine* layout

[3] http://www.dcm4che.org

- Plugins for the *p-medicine* Security Framework
  - Identity Consumer ext-plugin (for Single Sign-On)
  - Login portlet
- Plugins for Data Mining tools
- ObTiMA
- Template site for creation of a default *p-medicine* community
- Ontology Annotator

5.1.2 Ontology Annotator tool

The Ontology Annotator is a web-based tool aimed at end-users who want to add their databases in the p-medicine infrastructure. It is accessible through the project portal and offers a graphical interface for creating database annotations that enable the automatic translation of the respective database into HDOT format. More details about the Ontology Annotator tool can be obtained in deliverable D4.3.

The tool is completed to a great extent, although development is still on-going. Most planned features have already been included. Most planned features have already been included. New features not planned at the beginning have been added in order to foster tool usability and accessibility (namely, the ability to dynamically select HDOT modules and the ability to share annotation projects with other users by submitting them invitations) and internal tests with fake data have already been performed. In the upcoming weeks, tests with real data from the CDS scenario will take place and the final features will be implemented. The tool will be demoed in the 2nd project review.

The Ontology Annotator tool has interfaces with five components of the p-medicine architecture:

- The portal, acting as container of the Ontology Annotator tool and allowing end-users to access it,
- The Data Warehouse, since indexes of each database annotation generated with the Ontology Annotator tool are submitted to this component of the architecture,
- The database triplifier, as this component is in charge of providing RDF versions of any external database (either relational, Access, Excel or CSV databases are supported) which can act as input for the Ontology Annotator tool,
- The Security Infrastructure, allowing the Ontology Annotator tool to ensure data access restriction,
- The HDOT ontology, by representing it graphically and allowing users to annotate their databases based on HDOT.

The storage of the actual database annotations are currently done in the Ontology Annotator itself, however this might change in the future and the storage of these elements might move to the cloud storage. Nevertheless, this will not be performed before the 2nd project review, to simplify the development until this event.

5.1.2.1 Integration status

5.1.2.1.1 Portal

The tool is already accessible through the project portal, under the “tools” tab. The tool itself is maintained in a server at UPM’s facilities. The portal simply links to our server, which we regularly update with new improved versions. This approach allows simplifying integration with the portal, so no more integration steps with this component will have to be performed. The functional integration in this case, as mandated from the p-medicine architecture, is achieved by offering access to the Ontology Annotator tool through the portal. This has already been fully achieved. Regarding the semantic integration, no semantic integration is needed at this level.

5.1.2.1.2 Data Warehouse

The tool is also integrated with the Data Warehouse. The database annotations produced with the Ontology Annotator tool are used by the Data Warehouse in the process of translating databases into an HDOT-compliant format. Therefore, the Data Warehouse needs to be aware of these annotations. To achieve this, the Ontology Annotator tool will use the REST interface of the Data Warehouse. Each time a user creates or modifies a database annotation, the Ontology Annotator tool will use the REST interface of the Data Warehouse to properly notify this event. This task has not begun yet, and is planned to start within the next two weeks. The task itself will not require many man hours, since the REST interface of the DW is already available, and the details of the notifications are already agreed between the two development teams (UCL and UPM). No semantic integration is required at this level. The provided data to the DW follows our own annotation standard, but the Data Warehouse does not need to parse it or analyse it in any way.

5.1.2.1.3 Database Triplifiers

The database triplifiers component, developed by UCL, allows generating RDF-equivalent versions of any database submitted by end-users. This is needed as a first step in the annotation process, since the Ontology Annotator requires RDF databases to work. This component is provided to UPM as a Java API which is locally invoked to perform the translation. The API has already been integrated in the Ontology Annotator, and is correctly invoked by UPM’s tool. Problems have arisen due to the triplifiers still not correctly treating all input sources, but that issue is localized to the UCL triplifying API. As for the semantic integration, the triplifying API produces RDF versions of any inputted database. The Ontology Annotator is already capable of correctly treating the produced RDF models and representing them in the graphical interface.

5.1.2.1.4 Security infrastructure

The Ontology Annotator tool must comply with the security requirements imposed in the project. In order to ensure restricted access to the handled data (database schemas), only registered users are able to access the tool. This is done by implementing the Single Sign on policy developed by Custodix. This policy has already been integrated in the Ontology Annotator without any issues to report. The functional integration of the Single Sign-in policy of Custodix is currently implemented in the Ontology Annotator. The process involved the parsing of incoming HTTP requests, which was already achieved. No semantic integration is required at this level.

5.1.2.1.5 HDOT

The Ontology Annotator makes use of the HDOT ontology to represent it graphically to the end-users. This involves the parsing and subsequent analysis of the ontology contents. This process has already been achieved. Regarding its functional integration, our tool uses a local file version of the HDOT ontology. Therefore, the interface with this component refers only to the parsing of this file, which has been already achieved. As for the semantic integration, the interface with HDOT involves the analysis of the elements themselves of the ontology. Most

elements of HDOT are already integrated in the Ontology Annotator tool. Some minor work is left in order to retrieve annotation elements (natural language descriptions of the classes of HDOT) and their subsequent representation in the tool.

5.1.2.1.6 Complying with the views of the architecture

The integration of the Ontology Annotator tool with other components fully complies with the view of the architecture. Our tool also uses several third party libraries, namely:

- Jdom (http://www.jdom.org/)
- Jena (http://jena.apache.org/)
- OWLAPI (http://owlapi.sourceforge.net/)
- Raphaël (http://raphaeljs.com/)

5.1.2.1.7 Problems faced

There have not been important problems during the integration of the Ontology Annotator in the p-medicine platform. The integration in the portal was very simple and did not require any special efforts in terms of development of the tool. The integration in the Data Warehouse will take place in the upcoming weeks, but the interface is agreed with its developers, so no problems are expected. The integration with the security infrastructure has also been achieved with no issues worth mentioning. HDOT has also been integrated, but this formed part of the actual development of the Ontology Annotator. Big scale tests with real data are still left to do, so it is expected that some issues arise. Nevertheless, local tests have unveiled most important problems so far, and any issues left should be rather easy to solve.

5.1.2.2 Next integration steps

Next steps will be achieving full integration with the data warehouse, so the Ontology Annotator tool can submit database annotations to it. This will take place in the upcoming weeks. Tests with data from the second review scenario will also take place, allowing a complete evaluation of the tool behaviour.

Figure 6 shows a screenshot of the Ontology Annotator tool displayed inside the p-medicine portal.

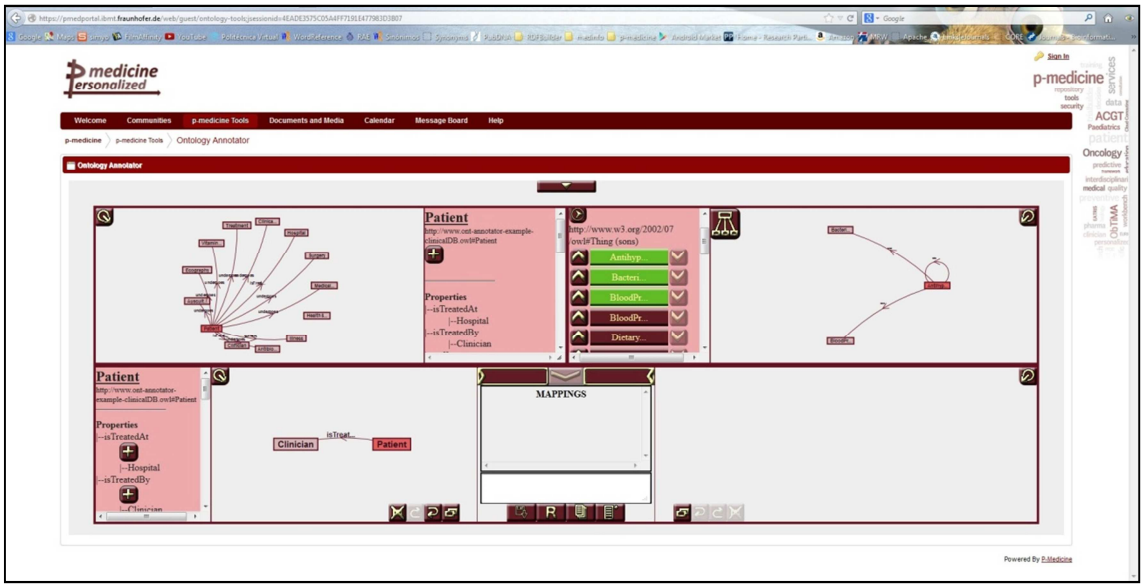

Figure 6 : The main Ontology Annotator window, in the p-medicine portal

5.1.3 Data Translation service

The Data Translation service is a Java-based API in charge of performing the translation of external databases into HDOT format. To achieve this, this API takes as input the external database in RDF format and a proper database annotation. The output is the same data, but complying with the semantics of HDOT. More details about the Data Translation service can be obtained from deliverable D4.3.

The implementation of the Data Translation service began in late December 2012. The basic functioning of the tool is already developed, and tested with sample fake data. The API is already capable of translating databases into HDOT-form. The missing implementation refers to the actual integration of the translated data into the Data Warehouse.

5.1.3.1 Integration status

The Data Translation service interacts with two components of the p-medicine infrastructure:

- The Data Warehouse invokes this service locally, allowing it to obtain the data in terms of HDOT,
- The HDOT ontology, as the Data Translation service must produce a new database containing the same data in terms of HDOT.

5.1.3.1.1 Data Warehouse

Integration with this component has not begun yet. Currently, the two teams involved (UCL and UPM) are discussing the details of the interface. The integration is expected to be completed throughout February 2013. The Data Translation service is a java based API which will be employed by the Data Warehouse. The interface call will therefore be simple code invocations, which simplifies the process. The actual form of the calls is yet to be agreed. HDOT will be the format in which the data translation service will provide the results to the Data Warehouse. HDOT is already adopted by the Data Translation service in order to produce the results that will be integrated in the Data Warehouse.

5.1.3.1.2 HDOT

The integration of this component in the Data Translation service has already been achieved. The ontology is used to build an HDOT-compliant version of the provided database. Similarly to the case of the Ontology Annotator tool, the functional integration consists on the parsing and analysis of a file containing the latest version of the HDOT ontology. In this case no annotation elements from HDOT are necessary. All required elements from HDOT are already properly used in the Data Translation service.

5.1.3.2 Complying with the views of the architecture

The integration of the Data Translation service with other components fully complies with the view of the architecture. Also, several third party libraries are used, in share with the Ontology Annotator tool:

- Jdom (http://www.jdom.org/)
- Jena (http://jena.apache.org/)
- OWLAPI (http://owlapi.sourceforge.net/)

The full integration of the Data Translation service in the p-medicine infrastructure will take place in the upcoming weeks. No important problems are envisaged at this stage, as the details of the interface with the Data Warehouse are being currently agreed.

5.1.3.3 Next integration steps

Next steps will be achieving full integration with the Data Warehouse, so this component can perform the required database translations. The Data Translation service will integrate as an API invoked locally in the Data Warehouse, so communication will be very simple. Discussions are on-going to agree on a format of exchanged data (options are to use a set of triples or a Jena OntModel object for the external database and the translated database). After agreement is reached, implementation should happen in a short lapse of time, producing a functional API that can be tested in a real scenario.

Figure 7 shows the class diagram of the Data Translation service.

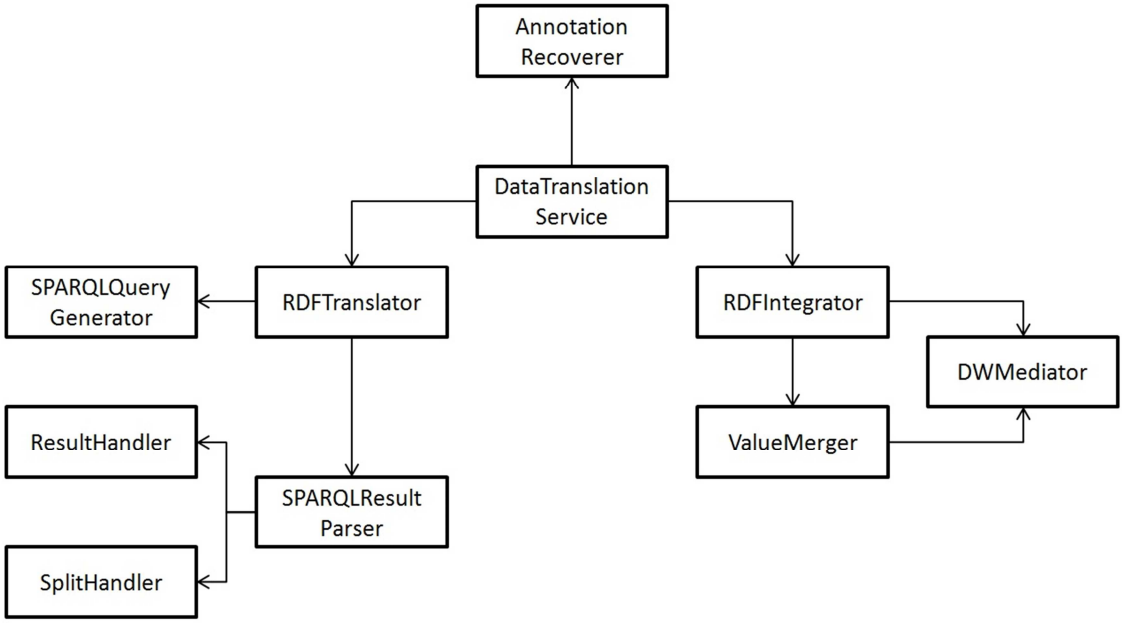

Figure 7 : The class diagram of the Data Translation service

5.1.4 Ontology Aggregator tool

The task of the Ontology Aggregator Tool (OAT) is to assist the user in the description and annotation process of biomedical data. By annotating the data the user applies concepts that are provided by the Health Data Ontology Trunk (HDOT)[4] middle-layer ontology or by pre-existing modules for it. Middle-layer ontologies do not specify semantic class or concept descriptions in great detail but rather contain classes and concepts on a more general level of domain representation in order to facilitate data integration and semantic interoperability. For specific purposes middle-layer ontology must be specialized in simple dedicated

4 http://code.google.com/p/hdot/

modules. For this reason we foresee that in the data (schema) annotation process HDOT or pre-existing modules cannot provide a suitable concept for the description of data. At this point the OAT comes into play so that the user can add the needed semantic description to HDOT and construct a specialisation in a module for it. Independently of this, the OAT should also be accessible via the p-medicine portal.

The OAT is still under development, but the core components have been implemented. Regarding its integration with the p-medicine platform, OAT mainly interacts with the ontology annotator tool developed by UPM. The OAT is triggered by an annotator tool failure. The data exchange is mainly in the form of raw text but we also check for possibilities to exchange data from the user's portal login. The OAT is fully integrated with HDOT and its modules.

Because the OAT can be triggered by any data schema annotation failure it can occur following any data schema annotation scenario. The OAT uses NCBO's BioPortal[5] as a data source and library of semantic standards and as a result, a problem which is sometimes faced is that BioPortal's rest interface for retrieving data is sometimes unreliable. Next step in its implementation roadmap is to develop a GUI together with UPM in a similar look and feel as the annotator tool uses. After this is finished, the integration into the portal can be done.

For illustration purposes, below is a diagram of the components. The components potentially interacting with the annotator tool and the p-medicine portal are PreprocessorImpl, PreferenceManager and loginPlatform.

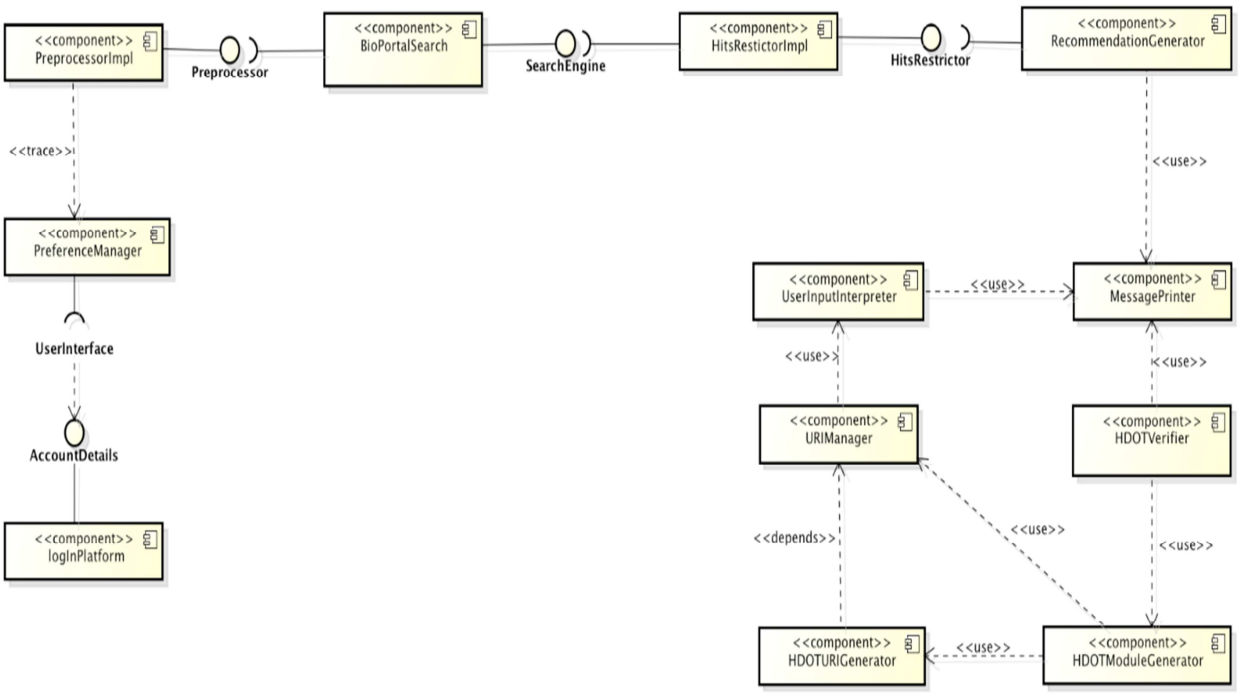

Figure 8 : Diagram of the components

5 http://bioportal.bioontology.org

5.1.5 Privacy framework

The goal of the pseudonymisation platform is to de facto anonymise the data before delivering to the p-medicine research domain. All data that is passed to the p-medicine platform must therefore pass through a state of the art pseudonymisation platform. This implies that all tools and software components that import data to the research platform must integrate with the pseudonymisation platform.

The pseudonymisation platform consists of two rounds of de facto anonymisation. A first round of de facto anonymisation is performed within the boundaries of the treatment domain. Afterwards the data is passed to a Trusted Third Party (TTP) that performs a second round of de facto anonymisation before delivering the data to the p-medicine research domain. CATS (Custodix Anonymisation Tool) will be used as default pseudonymisation tool. However hospitals are free to choose whatever anonymisation tool they want to use, as long as it has been approved by the CDP.

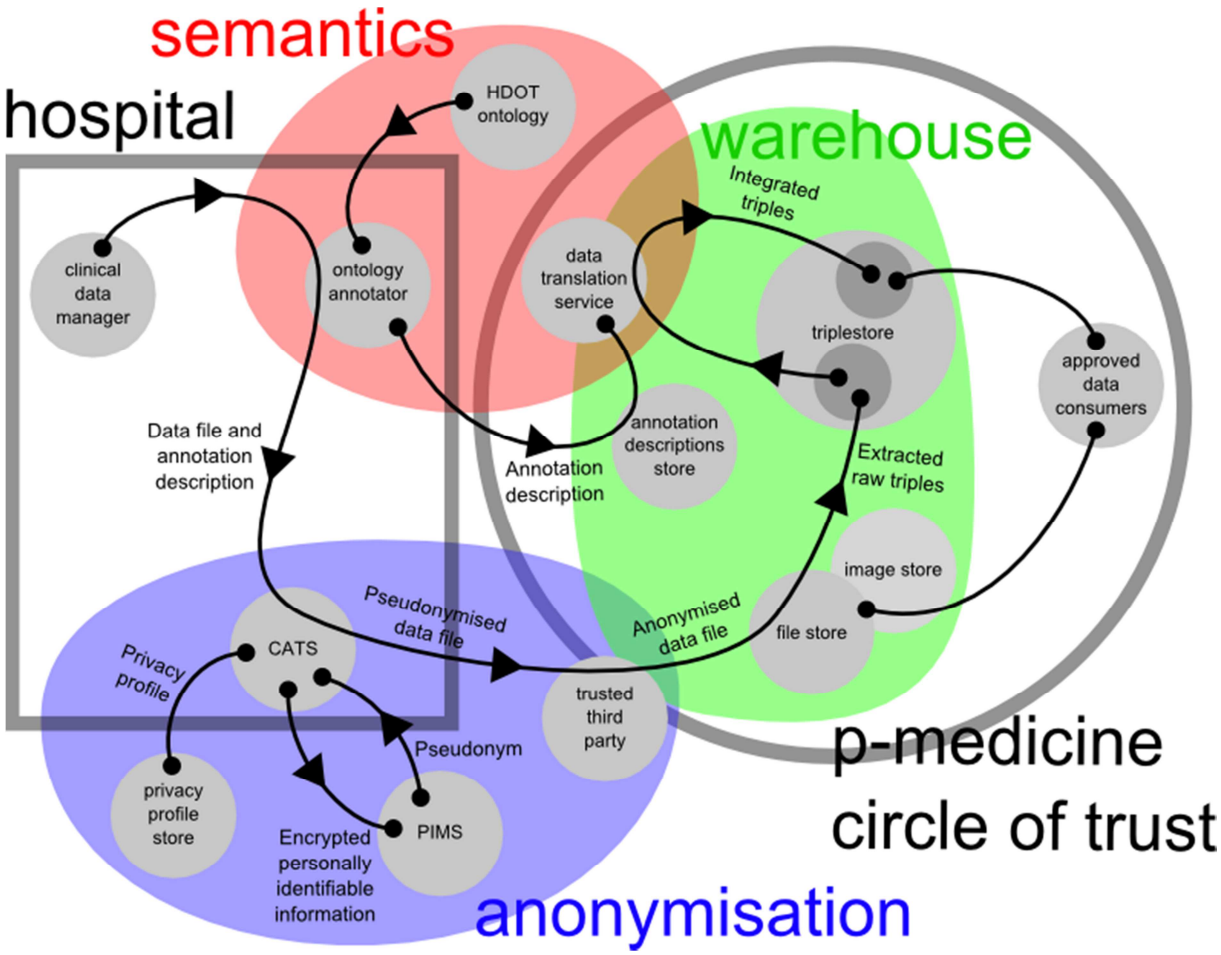

Figure 9 : Conceptual architecture of p-medicine data warehouse

The figure above depicts the conceptual architecture of the p-medicine data warehouse and its key components. Data (clinical, molecular, DICOM...) from the hospitals flows to the data warehouse, passing through the anonymisation domain. Data files are uploaded to CATS. CATS' task is to de facto anonymise the data files it receives. This means CATS has a public interface, and tools within the treatment domain can integrate with it. CATS closely interacts with PIMS, the patient information management system. In short PIMS' responsibility is twofold:

- To store a link between the personally identifiable information and the patient pseudonym.
- To link patient records representing the same real-world person, coming from different sources, to one index. This is called building a Master Patient Index (MPI).

After CATS processed a data file it will be sent to the TTP for a second round of pseudonymisation to complete the de facto anonymisation. The TTP will transform all pseudonyms in a given data file to a p-medicine specific pseudonym. Re-identification of this pseudonym is only possible by passing through the TTP again.

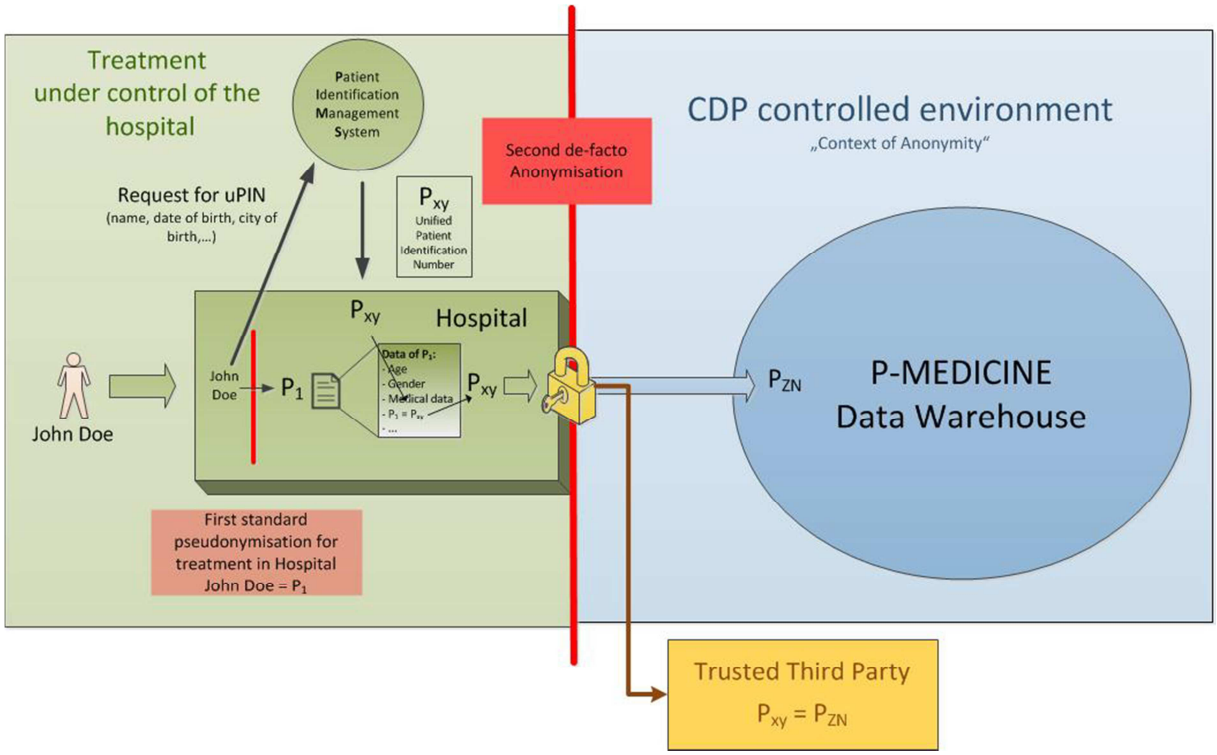

Figure 10 : Two rounds of de facto anonymisation

In the figure above a schematic representation of the two rounds of de facto anonymisation is shown. The first round of de facto anonymisation is performed in the hospital. Patient identifying data is removed from data files and replaced by a pseudonym. Data files are passed to the TTP that in turn performs a second round of de facto anonymisation.

5.1.5.1 Privacy Framework Components

The following components make part of the privacy framework:

- CATS (Custodix Anonymisation Tool Services) which is responsible for the de-identification or anonymisation of (clinical) data files.
- PIMS (Patient Identity Management System), a personal information management system, tracks patient identifiers originating from different data sources.
- CAT (Custodix Anonymisation Tool) is a tool through which the privacy profiles used by CATS can be created.

These components are described in more detail in *D8.3 "Release and Demonstration on data anonymisation tools"*.

5.1.5.2 Integration with Privacy Framework

In previous section we gave a general overview of the anonymisation platform. Below we describe in more detail some scenarios in which CATS/PIMS may be used.

5.1.5.2.1 CATS within the hospital domain

If CATS is installed in the domain of a hospital, files containing patient data may be sent to CATS in plain text. The most straightforward way is to create a web front-end. A healthcare specialist can then select the file he/she wants to send to the data warehouse, and upload it to CATS for the first round of de facto anonymisation. Once CATS finishes the file processing it will deliver the processed file to the TTP for the second round of de facto anonymisation.

Although a web-based file uploading solution is quite straightforward, it might not be that handy when a batch of files must be send to the p-medicine research domain. Uploading files one by one might be a time-consuming task. Easier would be to have an application (CATS client) run on the user's local system. This client would then scan a folder for available files, process and upload them to the CATS server. The CATS client is further described in *D8.3 "Release and Demonstration on data anonymisation tools"*.

Furthermore, the CATS web front-end can be integrated into the Liferay portal.

5.1.5.2.2 CATS outside the hospital domain

If no instance of CATS can or may be installed within the domain of the hospital, the de facto anonymisation needs to be done by a client tool. Assuming the hospital uses the Custodix anonymisation tools, the CATS client is used by the healthcare specialist to de-identify the data.

While processing data files, the tool must integrate with PIMS in order to provide patient identifying data and request patient pseudonyms. After file processing the result file must be delivered to the CATS server for further processing (see next section).

The client tool can be implemented as a graphical tool or a simple command line tool. In both cases a user must be able to select a directory from the local file system to process multiple files in batch. In a graphical tool, that would be by choosing a directory using a file selector dialog, in a command line tool that may be accomplished by providing start up arguments.

The biggest disadvantage of using a client tool is the need for client side installation. In an initial phase the client tool must be rolled out over all client machines participating in the project, and of course during the life time of the project, the tool must be installed on new machines as well. A good solution is to use Java Web Start. This Java technique allows for a user to download a Java application and run it on his client, just by clicking a link from a web site. Using this technique, CATS may serve as the provider for the client tool as well as the provider for the CATS privacy profiles.

5.1.5.2.3 CATS Usage

5.1.5.2.3.1 CATS and PIMS

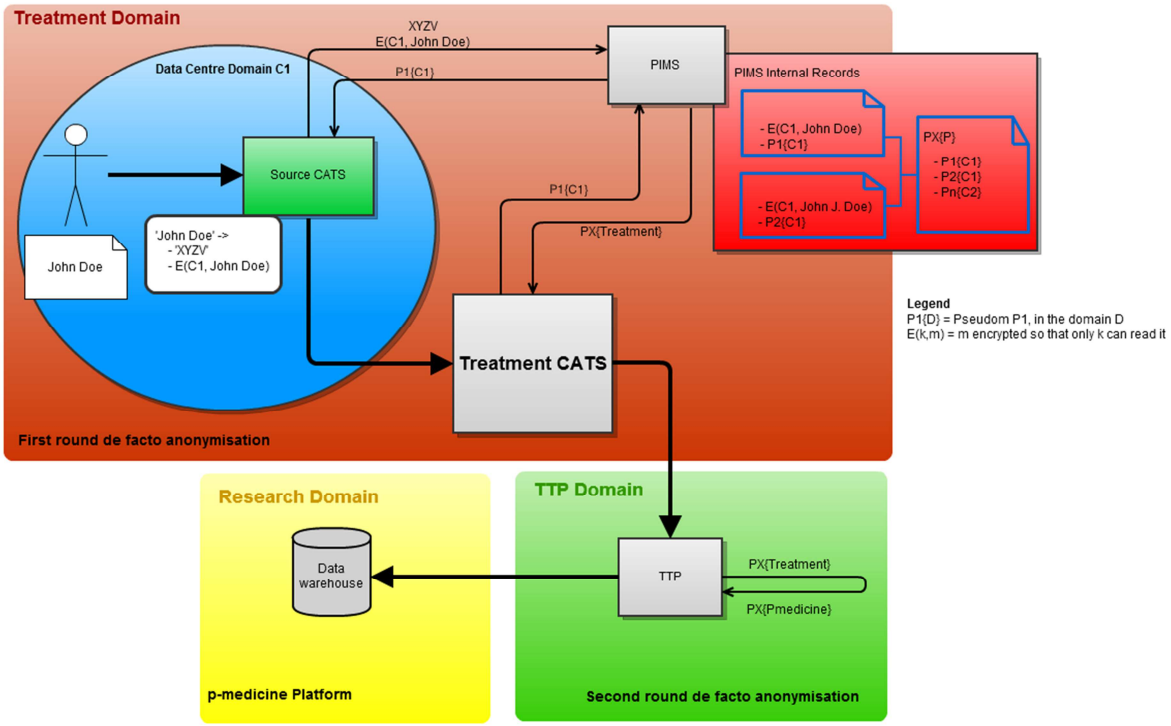

Figure 11 : De-identification architecture with CATS and PIMS

The figure above is an enhancement of the figure in deliverable *D5.1 (section 7.4.2.3)* which is conceptually more correct. It shows a more detailed schematic of the CATS process. In this scenario source CATS runs within the boundaries of the hospital. It pseudonymises data files by requesting source pseudonyms to PIMS that runs outside the hospital domain but inside the treatment domain. The source CATS (as is the source CATS' instances in other hospitals) delivers to the treatment CATS. This CATS instance will replace all source pseudonyms with treatment pseudonyms which are unique over the treatment domain. In turn the treatment CATS delivers its processed files to the TTP.

Furthermore PIMS receives patient identifying data from source CATS and keeps the link between pseudonym and patient data for later re-identification.

A given CATS instance will receive data files whether through its web interface or SOAP/REST web service interface. Depending on what integration interfaces and the types of data files that are processed, a different channel may be used to deliver. An instance of source CATS delivers to an instance of treatment CATS, which will most likely go through a secured web service. How the treatment CATS will deliver to the TTP is still to be defined.

Since PIMS is responsible for linking patient data and issuing treatment pseudonyms, it must run outside the hospital but within the boundaries of the treatment domain. This means patient identifying data will be encrypted before sending it to PIMS. That makes record matching even harder since encryption does not sustain the similarity between records. Record matching on encrypted data is described in deliverable *D3.4 "Service Integration Guidelines"*.

Section 2.1 describes CATS as a combination of a CATS client and a CATS server. In the above diagram source CATS is depicted in the domain of the hospital. However, the CATS server may run outside the domain and the CATS client may be used for file anonymisation at the source.

5.1.5.2.3.2 CATS without PIMS

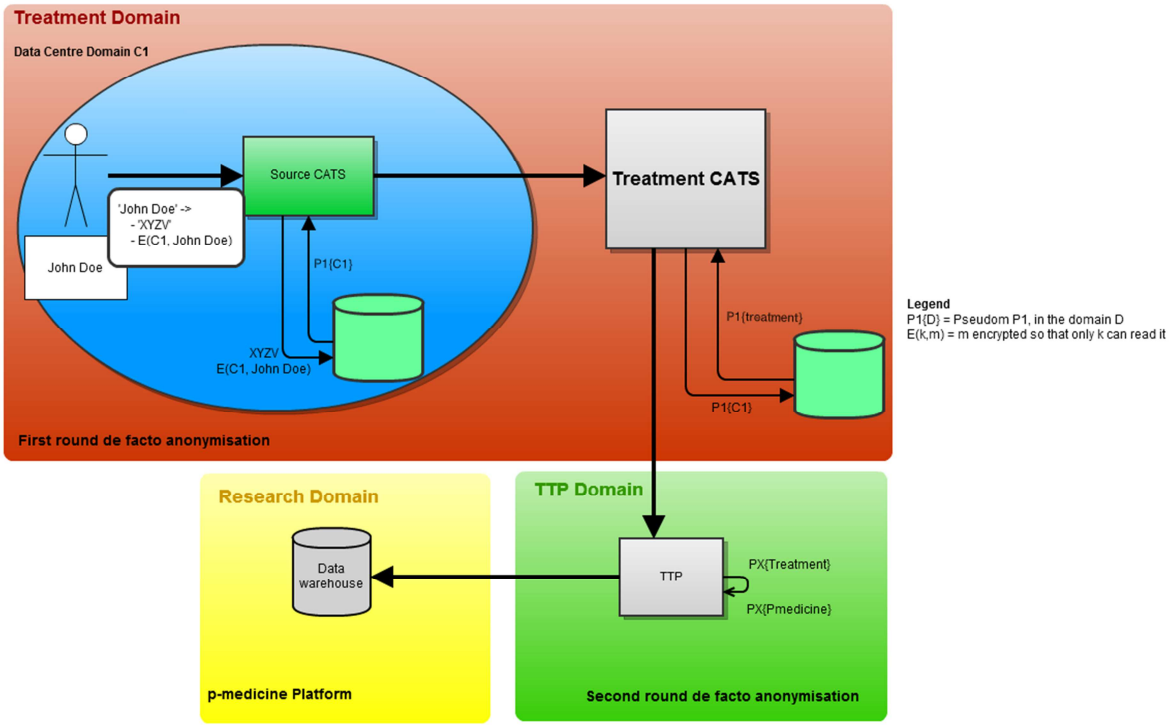

Figure 12 : De-identification architecture with CATS

The figure above depicts another scenario. In this case users upload data files to CATS in the same way as the first scenario. However it is CATS' responsibility to issue pseudonyms and store the reference back to the patient data. Since PIMS isn't used in this scenario, there will be no record matching.

Integration guidelines are the same as in the first scenario. Data files must be uploaded to CATS through its secured interfaces.

The source CATS server may again, run outside the domain of the hospital.

5.1.5.2.4 TTP

The Trusted Third Party is responsible for the second round of de facto anonymisation. It receives data files from the pseudonymisation platform and scans them for treatment pseudonyms that are replaced by new, p-medicine specific pseudonyms. The link between the pseudonym originating from the anonymisation platform and the new pseudonym is kept at the TTP. This means the de-identification of a p-medicine pseudonym must go back through the TTP. The exact de-identification flow still needs to be discussed.

5.1.5.2.4.1 Output channel

Eventually de facto anonymised data files are delivered to the p-medicine data warehouse. How the TTP will integrate with the data warehouse depends on what interfaces the data warehouse will provide.

5.1.6 Security Framework

The p-medicine security framework is designed around the SAML standard. A p-medicine Identity Provider (IdP) provides identity assertions to all services within p-medicine.

Web sites integrate with the IdP by providing a SAML compliant Identity Consumer (see *D3.4 "Service Integration Guidelines"* for more details). An Identity Consumer consumes and validates the assertions provided by the IdP.

REST Web Services integrate by accepting a SAML Identity Assertion through the HTTP Authorization header (see *D3.4* for more details).

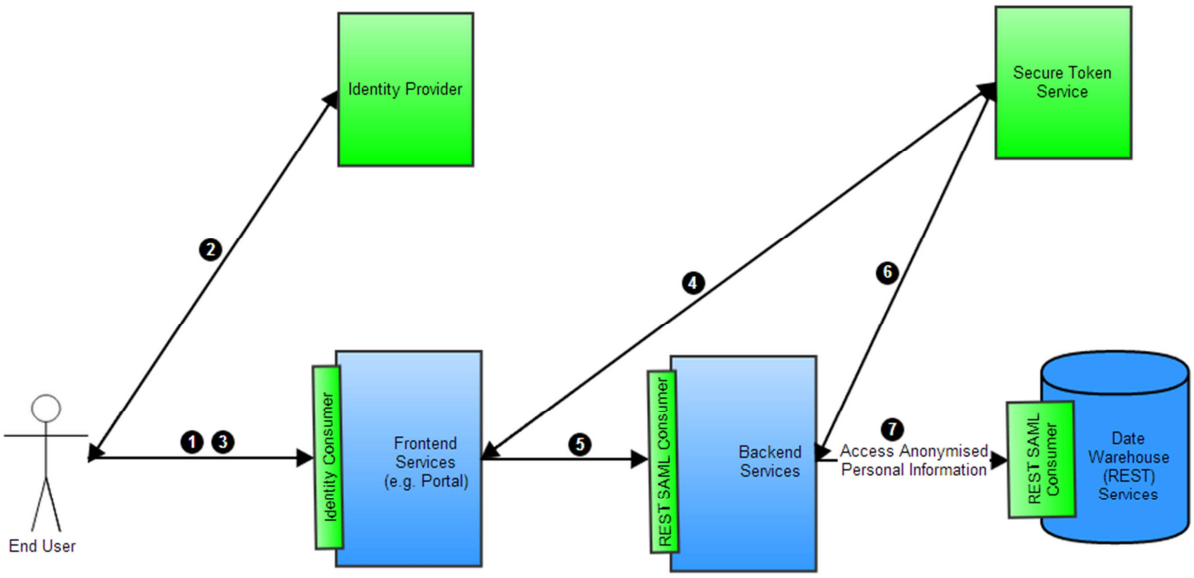

Figure 13 : Authentication Flow

Figure 13 shows authentication as defined in *D3.4* in a typical service call chain. A user (1) is redirected to the IdP when he visits the p-medicine portal. The user authenticates himself (2) on the IdP and is then redirected (3) back to the portal passing through an identity assertion. This assertion identifies the end user. Through a portlet the user then wishes to call a backend service. For this the portal requests a delegation token from the Secure Token Service (4). This delegation token contains the identity of the portal, which acts as intermediate service, and the end user on whose behalf the portal is acting. The delegation token is then passed through a HTTP Authorization header of the REST call (5) to the backend service. This backend server in turn also requests a delegation token to call the data warehouse (6 & 7).

5.1.7 Data Warehouse

The Data Warehouse is the logical repository of data in the p-medicine architecture. It relies upon cloud storage for the physical storage of files and objects and it provides various interfaces for inserting or reading data from it. Its design and implementation is described in detail in deliverables D7.1 and D7.2.

5.1.7.1 Implementation status

A SPARQL query interface for querying a data warehouse triplestore has been implemented as a Liferay portlet. It allows free text entry of a SPARQL query, which is evaluated, and the results displayed in an HTML table. The display is paged so that many results can be browsed.

5.1.7.2 Integration status

The portlet is integrated with the data warehouse triplestore hosted at PSNC, which it accesses through the REST programmatic interface via HTTP using a SAML token from the Custodix AAI system. It is a Liferay portlet, deployed on a local Liferay server at UCL, and which is easily deployed on the main p-medicine portal and integrated with the single sign-on service.
The portlet makes use of data from the warehouse, which will ultimately use the semantic resources from HDOT and other ontologies. In the future, particularly with the faceted browsing and commenting/rating facilities described below, it will make more direct reference to particular terms in ontologies.

No problems have been encountered implementing or integrating this tool so far.

5.1.7.3 Next steps

The tool so far developed represents the "Advanced search page" described in Section 3.7.4 of Deliverable 7.1. This is the minimum querying interface needed to support development of tools which rely on data in the warehouse, such as data mining and modelling tools. The remainder of the tool as described in Deliverable 7.1 must be implemented:

1. Free-text search
2. Faceted browsing of:
    1. Disease types
    2. Research institutions
    3. Others, to be decided
    4. Generic faceted browsing
3. Display of the latest changes to the structured data store
4. Display and addition of comments and ratings of data
5. News items relating to the running of the data warehouse
6. Help pages
7. Administrative contacts
8. Terms and Conditions for using the data warehouse, as signed by all data warehouse users, as a reminder of their responsibilities

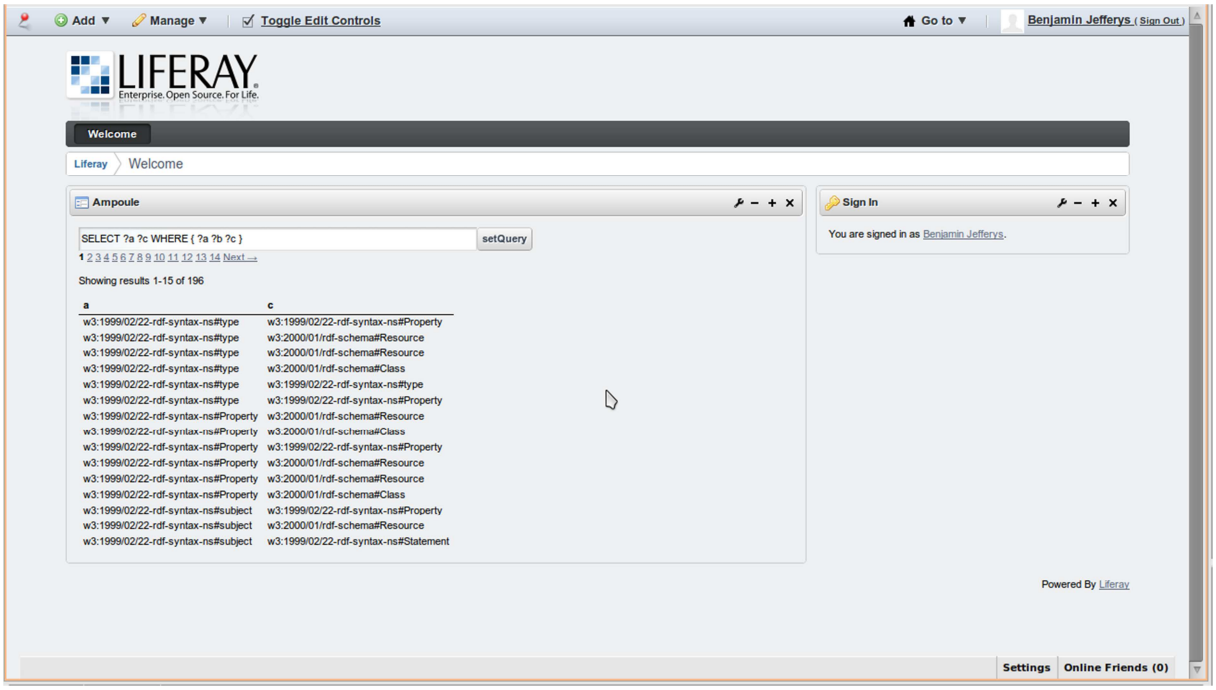

Figure 14 : Screenshot of data-warehouse browse and query tool

5.1.8 ObTiMA

5.1.8.1 Implementation Status

A detailed description of ObTiMA's current implementation status can be found in the deliverable D8.4 „Provision of new modules for clinical trial management" which has also just been submitted. Some of the main developments over the last period focused on improving the creation of Case Report Forms (CRF) and their subsequent use within a running trial or the introduction of a repository for storing and retrieving CRF templates (or parts thereof). The development of ObTiMA has progressed a lot over the last period and we are currently in the beta testing phase where the system is used to with real patient data by an experienced trial documentalist. After a successful testing, ObTiMA will be rolled-out to be used at the various clinical sites participating in the next SIOP study.

5.1.8.2 Integration Status

Most of the aforementioned developments affect only the core ObTiMA and thus is more or less independent of the p-medicine infrastructure. But there are several points of contact with the p-medicine infrastructure, namely:

- Since ObTiMA is based on an ontological foundation, the system employs the Health Data Ontology Trunk (HDOT) as its ontological fundament. By doing so this allows direct interoperability on the level of semantics between this system and other sources and targets such as the data warehouse.
- As the safety of all patient data is of utmost importance, the integration of the p-medicine privacy framework technology has been achieved as well and allows that patient data entered by the user can be encrypted „on-the-fly", on the user's client computer before being sent to the actual ObTiMA server.

- ObTiMA has also already been integrated into the p-medicine portal and can be started directly from within there. When doing so, ObTiMA is using the Single Sign On mechanism provided by the portal.

5.1.8.3 Architecture Compliance

The ObTiMA strictly follows the architectural principle of modular software design and development and by doing so directly follows the overall p-medicine as well. This means concretely that we are trying to partition the overall functionality of ObTiMA into smaller modules which interact via predefined interfaces. To implement this modularity, we employ the Spring Framework which provides functionalities to define and (later) configure modules in a standardized fashion. The modularization (or „decoupling“) also happens on the level of our internal data storage where we employ Hibernate as a data abstraction layer that removes the need to hardcode vendor-specific SQL. Further, the graphical user interface of ObTiMA is separated into a separate presentation layer which is based on the use of JavaServer Faces (JSF) and interacts with the data processing modules of ObTiMA. Finally, it is worth mentioning that in accordance with the other ontology-based tools within p-medicine, we are using the OWLAPI as well for all ontology-related implementations.

5.1.8.4 Faced Problem

We did not face any problem in our development which could be attributed to the overall infrastructure or architecture.

5.1.8.5 Next Integration Steps

In the subtask T8.5.2 „Sync service to facilitate the conduction of clinical trials through HIS connectivity“ a synchronization service will be developed to link ObTiMA to Hospital Information Systems (HIS). In particular, this service is intended to retrieve data items defined in the Clinical Record Forms (CRF) directly from the HIS of a given hospital. Such a service would then, at least partly, avoid the need to (re)enter data by hand into the CRFs in the case that those data are already stored in the HIS. To achieve this goal, their development will rely on the interfaces and implementations provided by the Ontology Annotator service and the Data Translation service: All data stemming from the HIS needs to be converted first from its original format (i.e. database schema) into a format that ObTiMA is able to understand. And since in ObTiMA CRFs can be ontologically annotated with classes from HDOT as well, the translation step mirrors the corresponding procedure when pushing data into the data warehouse directly.

5.1.9 Interactive Empowerment Services (IEmS)

The IEmS architecture shown in Figure 15 combines the progress of technology and biomedical research with patients' personal needs. Patients and doctors can access the Interactive Empowerment Service through the p-Medicine portal. Then, questionnaires and intelligent profiling mechanism are used in order to construct patient profiles. Thanks to collaboration among physicians, psychologists and ITs, the patient's profile can be combined with the Patient's Health Record System (PHR). PHR is one of the critical components of the IEmS, and where clinical information is patient-tagged and combined with his/her psychological, social and cognitive characteristics (patient's profile). Its importance is double: Firstly, it is a container of all the information related to the patient, such as information on diagnosis and biobank data as well as information on the possible clinical trials. Secondly, the patient his/herself has the opportunity to have access to such information. In the PHR, the patient can follow the journey of his/her data and based on it begin the decision process.

Furthermore, by having this information, the patient has the possibility to manage the consent for clinical trials and to track his biomaterials. By providing to the patients the ability of „consent management“ allows patients to control their own data and enhances the interaction between patients and doctors when a request for a new consent is needed, by increasing efficiency and again by involving the patient actively.

In the following we give a short description of the main building blocks and report their implementation status.

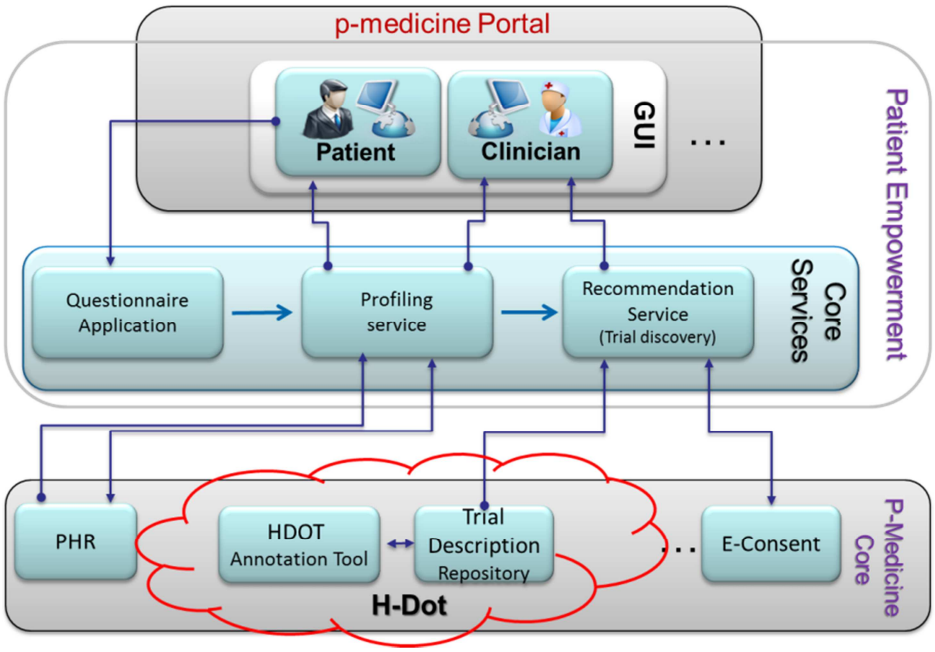

Figure 15 : High-level architecture of the p-medicine Patient empowerment services

5.1.9.1 ALGA – C Questionnaire

The ALGA-C questionnaire is used to collect psycho-cognitive information about patients. The combined clinical and psycho-cognitive information becomes patient's and doctor's common knowledge, around which they can build together long term efficient decision-making plans. Essentially, after the patient has completed the questionnaire his answers are stored in a database for further analysis. The analysis of the results is conducted in two phases. The first phase contains the calculation of simple average scores. These scores are calculated based on the fact that each question's answers have different weights. The second phase of the analysis contains the calculation of z-scores, which are more complex scores that take into account standard deviations of a healthy group of patients that have completed the questionnaire. The results of the analysis will be provided to the patient's doctor for facilitating their interaction in the decision making process.

Figure 16 : Screenshot of the ALGA-C Questionnaire from a mobile and a desktop platform.

Currently the questionnaire is fully implemented and the first part of the result analysis as well. As for the technical infrastructure of the application, the back-end is based on the Spring MVC 3 framework, Apache Tomcat server and MySQL database. Moreover, the interaction of the application with its database is facilitated by the Java Persistence API (JPA) EclipseLink. The front-end of the application is based on Java Server Pages, HTML5 and JQuery technologies. Moreover the application guarantees responsiveness by relying on the responsive CSS Framework, Twitter Bootstrap. Overall, the Questionnaire application is cross-browser (Chrome, Safari, Firefox, Opera, Internet Explorer) and cross-mobile (iPhone4, iPhone5, iPad, Android Nexus S, Android Table 7'', Android Tablet 10'') compatible. Finally, a patient module is developed in the HDOT ontology capable of representing the profiling information collected using this questionnaire. Figure 16 presents the usage of the ALGA-C Questionnaire from a desktop PC browser and a mobile iPhone emulator.

For the forthcoming year the second part of the analysis is planned. This is due to the fact that for this task healthy group data are required which were made available recently. Moreover, currently scenarios are being constructed to discover opportunities for linking the questionnaire to the p-medicine portal, for pushing data to the data-warehouse and for interconnecting it with the PHR system.

5.1.9.2 PHR System

Based on the outcome of an extensive evaluation of PHR systems[6], the IndivoX[7] is the PHR system that was selected and will be adopted in p-Medicine. This PHR system in extended for p-Medicine in several ways, such as an application that exports data in XML or JSON format and several forms that were missing have been implemented for data entry. For the time being scenarios are being developed that will demonstrate the adaptation of the user

[6] Irini Genitsaridi, Haridimos Kondylakis, Lefteris Koumakis, Kostas Marias, Manolis Tsiknakis, Evaluation of Personal Health Record Systems through the Lenses of EU Research Projects, Journal of Computer in Biology and Medicine (submitted).

[7] http://indivohealth.org/

interface according to individual patient profile. HDOT will be used for storing and retrieving profiling information to/from the central data warehouse. Moreover, the integration of the PHR with the rest of the p-Medicine platform and the portal is planned for the forthcoming year.

5.1.9.3 Recommendation Service

Currently, registering patients into clinical trials and finding eligible trials for patients require manual search and clinicians may be overwhelmed by the number of clinical trials and the exclusion and eligibility criteria.

However, having both PHR data and Trial Descriptions in the data warehouse p-medicine will allow the efficient recruitment of eligible patients for clinical trials. Recommendation service will use semantic matching to identify and provide suggestions of potentially eligible patients according to the available eligibility and exclusion criteria. This service is in the stage of initial implementations and we expect that it will be finished in the following months. By automatic matching, we expect to reduce the search space with respect to the number of patients, CTs and exclusion/inclusion criteria that need to be manually reviewed to approximately 20% of the original search space.

5.1.9.4 e-Consent

The patient's written informed consent is mandatory for research use of human biomaterial. "Multi-layered" consent, which requests from patients to make different choices on research that might or must not be performed on their samples, is increasingly recommended by ethics experts as a participative tool for patients. Management of multi-layered consent forms is to some extent already implemented in Biobanks Information Systems. For the time being, p-medicine is exploring the possibility to use the entire infrastructure from the CONTRACT project but it is not decided yet.

5.1.10 p-medicine Workbench

The p-medicine workbench is the end-user application that provides access to various p-medicine tools for the clinicians to use and invoke. This tool is basically an "application store" where different applications, tools, and services are categorized and indexed accompanied with information about their functionality, operation, development status, accessibility, etc. A screenshot of the Workbench user interface is shown in see Figure 17.

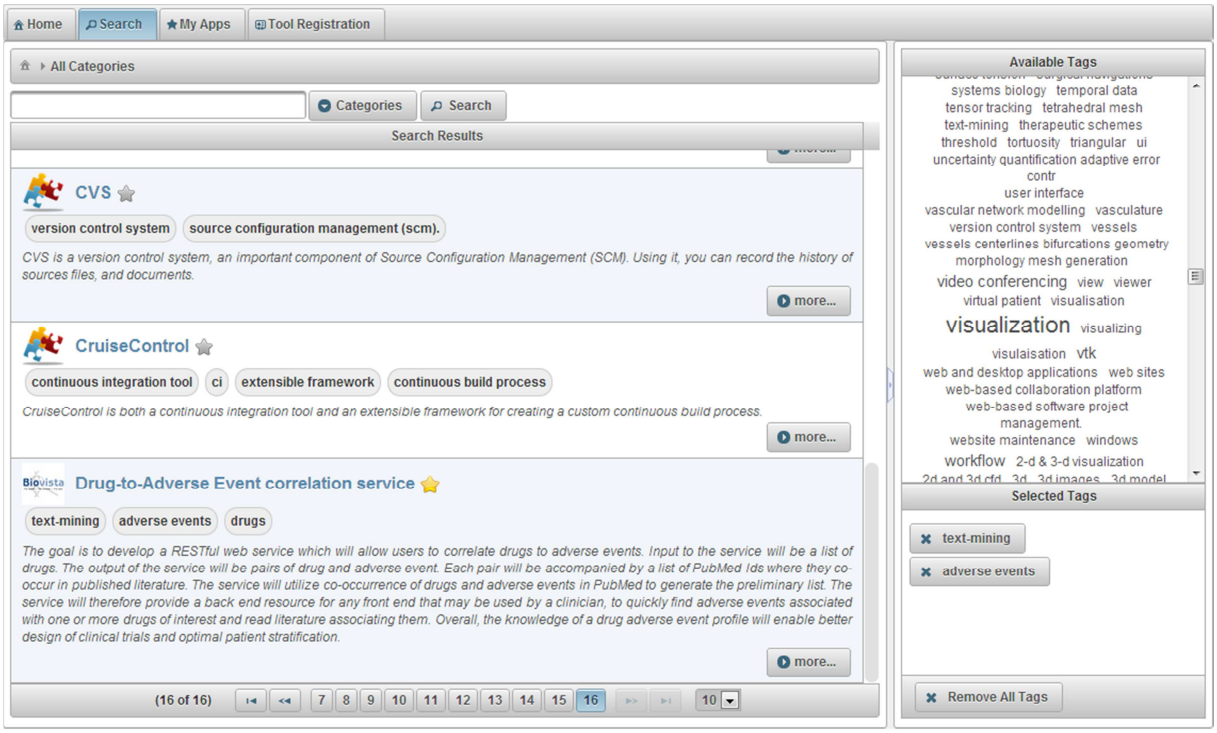

Figure 17 : Screenshot of the Workbench UI

The primary users of this application are the clinicians but of course it would be also useful to other user groups such as bioinformaticians etc. The requirements for the p-medicine workbench are the following:

- Provide information about potentially useful tools in the domain of clinical and biomedical cancer research.
- Support the various navigation and search strategies for the user to locate the most appropriate tool for a given scenario or clinical context
- Support for user personalization and customization based on the users behaviour and preferences
- Provide an intuitive search functionality not only for the discovery of the tools that are relevant to a clinical scenario but also for identifying pertinent background information e.g. for genes, diseases, and drugs by aggregating and linking available life sciences data sources.

The architecture of the workbench is depicted on the image below. As part of the p-medicine platform, the workbench features a portal integrated user interface that the users interact with. Additionally, the workbench as an application is built following a multi-tier architecture: Behind the scenes the workbench port let is supported by the workbench server as its “business logic tier” and a number of data sources and external tools. The p-medicine tools information is stored in an RDF “Triplestore” that supports intelligent search and inference in order to cater for complex search patterns.

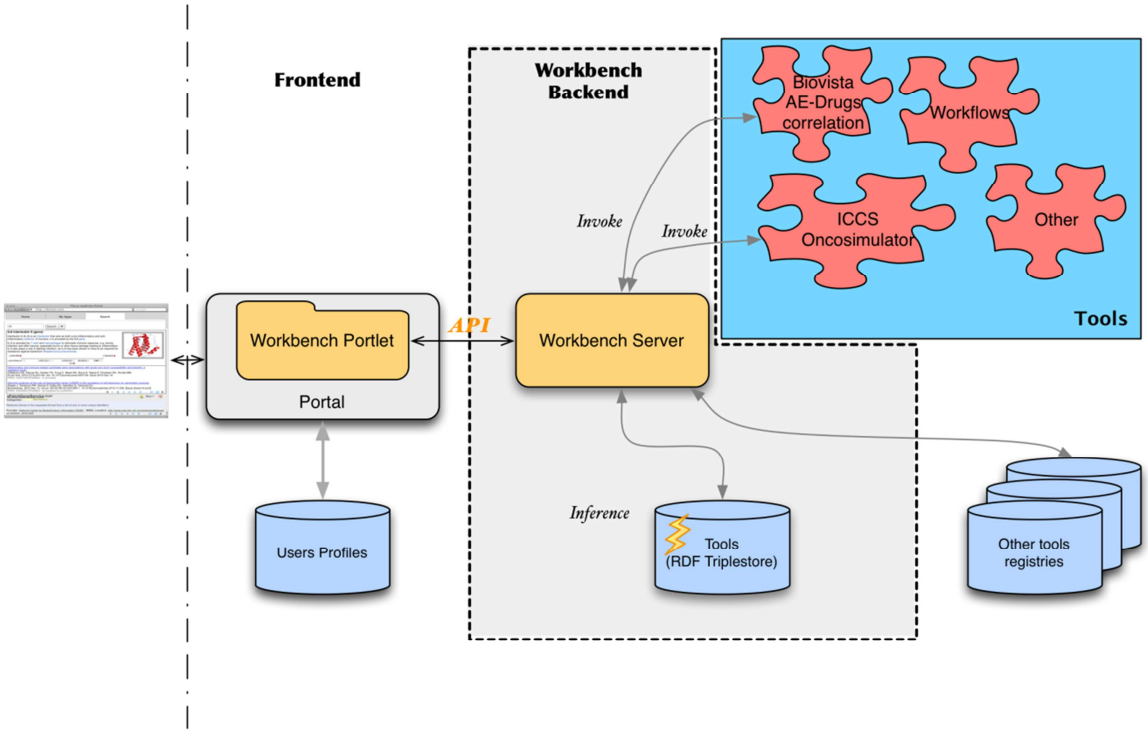

Figure 18 : The deployment architecture of the p-medicine workbench

5.1.10.1 Integration points

The workbench application is accessed through the p-medicine portal and therefore its presentation layer is integrated into the platform's user access and security framework. Additionally the workbench server contacts and invokes some key p-medicine specific tools, including, but not limited to the following applications:

- Adverse events - drugs correlation service
- Oncosimulator service
- Data Mining application and Workflows registry

For these interactions to take place it is important that the workbench application complies with the security and interoperability mechanisms adopted by the project.

As a provider of functionality, the workbench server will support a documented Application Programming Interface (API) to perform queries and support the navigation into the tools and data knowledge graph. This API is currently utilized by the front-end in order to contact the workbench server but in the future, when the envisaged functionality has been implemented and the API becomes stable, it will be offered for other p-medicine tools to use.

5.1.11 Cloud storage infrastructure

The cloud storage infrastructure which is behind the data warehouse is based on the OpenStack Object Storage (Swift) software. In brief, the features of this software are:

- Account/Container/Object structure. Within Storage Account user can create Containers (something like directories) and within them Objects are placed (files). Due to the performance issues, container nesting is not possible.
- Provides redundant, scalable object storage using clusters of standardized servers capable of storing very large amount of data
- Not a typical file system, but rather a distributed storage system for static data such as virtual machine images, photo storage (DICOM files), email storage, backups (DB backup files for example) and archives. Having no central "brain" or master point of control provides greater scalability, redundancy and durability.
- Objects and files are written to multiple disk drives spread throughout servers in the data centre, with the OpenStack software responsible for ensuring data replication and integrity across the cluster.
- Storage clusters scale horizontally simply by adding new servers. If a server or hard drive fails, OpenStack replicates its content from other active nodes to new locations in the cluster. Because OpenStack uses software logic to ensure data replication and distribution across different devices, inexpensive commodity hard drives and servers can be used. It is also possible to use any existing file systems which support extended attributes.

The hardware infrastructure on which this software is utilized comes from the PLATON[8] project. To adapt the Swift software to the p-medicine requirements, the following changes were made:

- To adopt p-medicine security system to file store OpenStack Keystone authentication and authorization system was used. It is a separate module for OpenStack software stack to centralize identity related processes with one central service. Additional module was created to support SAML assertions and to map recognized users to the storage accounts and users in OpenStack Object Storage. SAML module was installed and configured together with Keystone service.
- A bug inside Swift code was tracked which revealed inconsistency between object size and metadata size values in some specific use case. It was important as it is planned to integrate different file systems as backend storage in the future and caused many errors during preliminary tests. Patch prepared was accepted by the community and incorporated in next releases of the software.

5.1.11.1 Interaction with other components

There is a different level of integration of the cloud storage infrastructure with the rest of the p-medicine architectural elements. Currently it is used, or it is being integrated, with the following tools:

- Filestore is used by Data Warehouse as a file storage area.
- Will be used by p-medicine web portal to store user's files within private storage accounts, e.g. some final results of analysis of data
- Data mining module – uses filestore to save intermediate and final results during analyses of medical data

[8] http://www.platon.pionier.net.pl/online/?lang=en

- Is an archive service for the dc4chee DICOM storage server (DICOM files are archived on a regular basis inside cloud storage)

The assume based on the p-medicine architecture that it is very likely, future versions of other p-medicine components will also use cloud storage infrastructure as their physical storage facility.

5.1.11.2 Possible future work on the components

- Possible changes due to some extensions to security infrastructure of p-medicine, for example related to authorization actions and attributed exchanged in SAML assertions.
- Future integration with external file or data management systems like NDS[9] or iRODS[10]. As p-medicine project does not buy hardware it is planned to use existing infrastructures as storage backend. This action should allow satisfying growing demand on storage space.
- Web client application for Liferay portal, should allow end users to store files (related to the p-medicine domain) in the filestore directly.

5.1.12 Drug-to-Adverse Event correlation service & viewer

5.1.12.1 Implementation Status

A detailed description of this service and the user scenario it is based on can be found in deliverable D11.1. In short, the service allows users to correlate drugs with adverse events using published literature found in PubMed. It provide a back end resource for any front end, that may be used by a clinician, to quickly find adverse events associated with one or more drugs of interest and read literature associating them. Figure 19 shows the abstract idea implemented by the service for the specific instance of Drug to Adverse Event correlation.

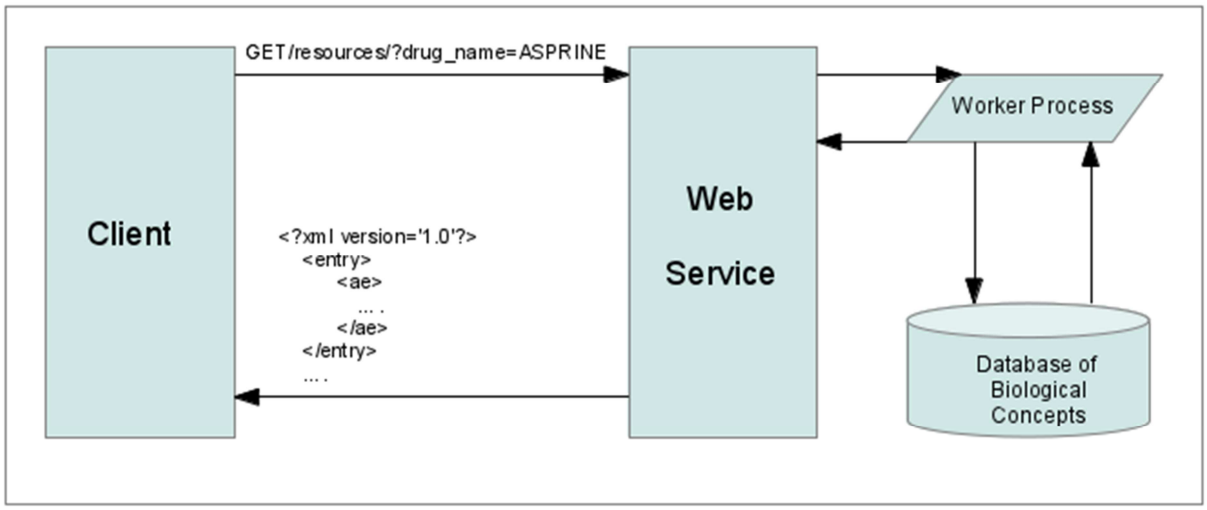

Figure 19 : Architecture of the Drug to Adverse Event correlation service

[9] http://nds.psnc.pl/
[10] http://www.irods.org/

The goal of the drug-to-adverse event correlation viewer is to develop a user friendly graphical frontend which consumes the Drug-to-Adverse event correlation service. The frontend provides a browser based client where the user will be able to specify values (drug name) for parameters accepted by the service. The result of the service will be displayed in the browser in two panels. The tool will provide a clinician a quick graphical means for finding adverse events associated with a drug and review related literature and demonstrate the plug-ability of the service (a requirement for its consumption in general user defined workflows). Preliminary implementation of the client is complete and integration pieces between the client and the service are in place. Work is continuing for incorporating further features to the client.

5.1.12.2 Integration Status

Implementation of the API of the service as described in deliverable D11.1 is complete and the software has already gone through one cycle of updates motivated by upgrades in the underlying web-service framework. The service provides a part of the functionality described in user scenario 7.1.3 (deliverable D7.1) and is therefore expected to interface with other p-medicine services / tools participating in the scenario.

5.1.12.3 Architecture Compliance

The service follows the overall architectural principal of p-medicine in that it is RESTful as well as it provides a very well defined functionality. Use of RESTful services in p-medicine is encouraged and deemed to be most suitable for building modular software. The web-service provides a RESTful API making it compliant with the overall architecture. Further, due to the extremely well defined functionality and XML interface the service fits well with the modular software design desired in the project. The viewer also follows the basic architectural principal of modularity in terms of the functionality it implements. However, the viewer is not intended to be incorporated into workflows as a step in the chain but rather as an endpoint for viewing information in an organized fashion. A sample drug search (Methotraxate) using the viewer and visualization of the results from the service is shown in Figure 20.

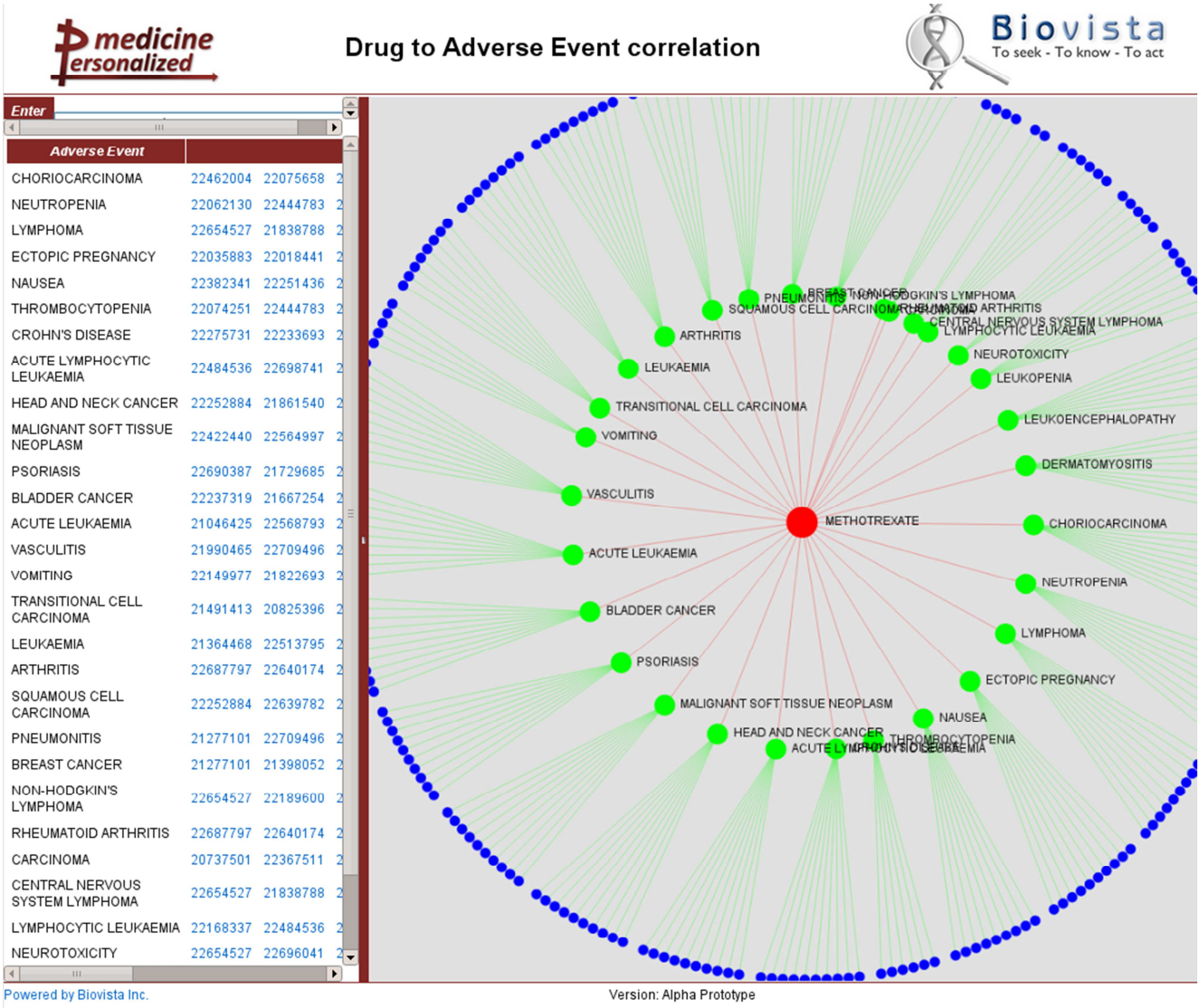

Figure 20 : Screenshot of the Drug to Adverse Event correlation viewer

5.1.13 Data Mining services

A data mining task in p-medicine can be described as workflow or as workflow pattern. A data-mining workflow is executable in the p-medicine Data Mining Webapp and can invoke any of the p-medicine data mining services and computational execution environments. A workflow pattern is not executable and serves as a template for data mining workflows. It provides rules and consistency checks for testing whether a workflow, when drawn from a workflow pattern, has been filled correctly. A detailed description of the Data Mining patterns and services is documented in the deliverable D11.1 “Initial Definition of Data Mining Patterns”.

The Data Mining Webapp component consists of an extensible collection of web-services which provide their functionality through a uniform service interface that can be accessed from the p-medicine portal. The general architecture is depicted in Figure 21.

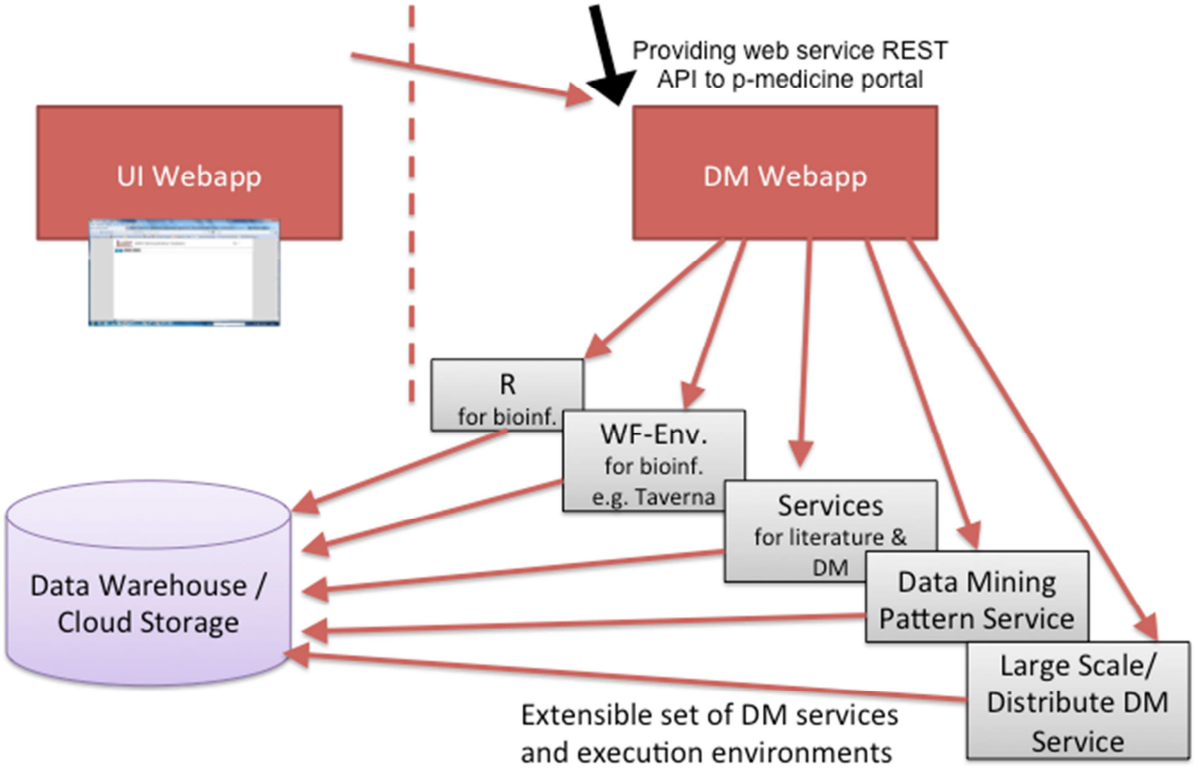

Figure 21 : Overview on the data-mining service architecture in p-medicine

This architecture integrates an extensible set of services providing computational execution environments for data-mining algorithms. As a main result of the user requirements analysis carried out in WP2 with respect to data mining patterns and services, it turned out that the users of data mining services in p-medicine, i.e. bio-physicians and data miners, have a major interest in being able to re-use existing workflows and algorithms and continue working with existing, possible different computational execution environments. In particular, development and re-use of algorithms encoded in the statistical language R and development and re-use of workflows for the Taverna workflow engine are initial requirements for p-medicine.

The basic purpose of the Data Mining Webapp is to provide a uniform web REST API to the p-medicine portal for accessing the individual data mining services. The Data Mining Webapp will forward the submitted data mining tasks to the data mining services lying “behind” the Data Mining Webapp. These services themselves can, but are not required to implement the asynchronous REST style described in D11.1. In case they just provide synchronous invocations, the Data Mining Webapp map organizes the REST style invocations for them.

5.1.13.1 Integration status

The Data Mining WebApp is being integrated into the p-medicine core architecture and will be accessible from the p-medicine portal. For making intermediate tests and demonstrations, a Data Mining User Interface has been developed as a separate Web Application, based on the Liferay Framework.

6 Demonstration scenario

Here we summarize an end-to-end workflow for p-medicine, which will be used as a demonstration scenario in the second annual review of the project. This scenario will demonstrate the initial integrated platform and the working prototypes of various p-medicine components.

The workflow will demonstrate how data can be exported from a hospital database, imported into the p-medicine framework, stored in the data warehouse and analysed with meaningful biomedical workflows, within a secure infrastructure with appropriate user interface. The workflow can be viewed as the concatenation of two separate workflows:

- The **IN-Workflow** imports data from hospital database into the data warehouse, where the data reside in a meaningful way in the p-medicine infrastructure.
- The **OUT-Workflow** reads data from the data warehouse and computes a meaningful result which can be exported or presented to the user.

The participating components to this demonstration are:

- **Push services**

  Takes input from a hospital database and inserts data into the data warehouse, following a flow of various processes such as anonymization and ontology annotation.
- **Ontology Annotator**

  Takes as input the database schema and shows how users can create adequate annotations for it. The generated annotations will be based on the semantic equivalences of the database schema with the HDOT ontology. The resulting database annotation will be uploaded into the data warehouse and subsequently used in the translation of the database into an HDOT-compliant format.
- **Data Warehouse**

  Stores the data it receives from the push service and the schema it receives from the ontology annotator onto the cloud storage.
- **Cloud storage**

  Physically stores the data files from the push service that are inserted into the data warehouse.
- **Data Mining services**

  Receives data from the cloud storage through the data warehouse interface and executes a realistic biomedical workflow.
- **Portal**

  Provides the user interface to all components that need a user interface and creates for the user an integrated and coherent view of the platform.
- **Security framework**

  It is used in various scenes of the scenario

This scenario does not intend to cover all aspects of the p-medicine platform or to exhibit every bit of detail of each p-medicine component, which are analysed in the

corresponding deliverables, but rather to demonstrate an initial version of the integrated platform and highlight the overall functionality that p-medicine aims to achieve.

Appendix 1 - Abbreviations and acronyms

ALL	Acute Lymphoblastic Leukaemia
API	Application Programming Interface
CAT(S)	Custodix Anonymisation Tool (Services)
CDP	Centre for Data Protection
CDS(S)	Clinical Decision Support (System | Service)
CRF	Case Report Forms
CSS	Cascading Style Sheets
CT	Clinical Trial
DICOM	Digital Imaging and Communications in Medicine
DW	Data Warehouse
HDOT	Health Data Ontology Trunk
HIS	Hospital Information System
HL7	Health Level Seven
HTTP	HyperText Transfer Protocol
IdP	Identity Provider
IEmS	Interactive empowerment services
JPA	Java Persistence API
JSF	Java Server Faces
JSON	Javascript Object Notation
LOINC	Logical Observation Identifiers Names and Codes
MPI	Master Patient Index
MVC	Model-View-Controller
OAT	Ontology Aggregator Tool
ObTiMA	Ontology-based Trial Management Application
p-BioSPRE	p-medicine Biomaterial Search and Project Request Engine

PHR	Personal Health Record
PIMS	Patient Identity Management System
RDF	Resource Description Framework
REST	Representational State Transfer
SAML	Security Assertion Mark-up Language
SIOP	International Society of Paediatric Oncology
SMART	Substitutable Medical Apps and Reusable Technologies
SNOMED	Systematized Nomenclature Of Medicine Clinical Terms
SOAP	Simple Object Access Protocol
SPARQL	SPARQL Protocol and RDF Query Language
TTP	Trusted Third Party
VPH	Virtual Physiological Human
XML	Extensible Mark-up Language

Appendix 2 – List of figures and tables