Deliverable 3.1: State-of-the-art report on standards

Grant Agreement Number:	270089
Deliverable number:	D3.1
Deliverable name:	State-of-the-art report on standards
Contractual submission date:	31/08/2011
Actual submission date:	31/10/2011

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.:	D3.1
Document name:	State-of-the-art report on standards
Nature:	R
Dissemination Level:	PU
Version:	0.4
Actual Submission Date:	31/10/2011
Editor:	Manolis Tsiknakis
Institution:	FORTH
E-Mail:	tsiknaki@ics.forth.gr

R=Report, P=Prototype, D=Demonstrator, O=Other

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)

Abstract

This document gives basic guidelines on how to evaluate standards, and then summarises the VPH toolkit guidelines, and standards for software engineering, HCI and world wide web, data exchange formats and protocols, security, interoperability, data warehousing, identity, biobanking and data mining.

Keyword list

standards, guidelines, protocols, formats, IT, state-of-the-art, review

Modification control

Version	Date	Status	Author
0.1	13/09/2011	Draft	Benjamin Jefferys
0.2	15/09/2011	Draft	Benjamin Jefferys
0.3	04/10/2011	Draft	Benjamin Jefferys
0.4	24/10/2011	Final	Benjamin Jefferys

List of contributors

- Benjamin Jefferys, UCL
- David Chang, UCL
- Giorgos Zacharioudakis, FORTH
- Lefteris Koumakis, FORTH
- Stelios Sfakianakis, FORTH
- Vangelis Kritsotakis, FORTH
- Elias Neri, Custodix
- Helmut Opsomer, Custodix
- Kristof De Schepper, Custodix
- Stephan Kiefer, FhG-IBMT
- Axel Poigné, FhG-IAIS

Executive summary

This document describes the relevant standards for information technology in the p-medicine project, with a view to separately establishing which might be adhered to, which might be contributed to or influenced, and which should be disregarded from the point of view of this project. It is a snapshot of an ongoing Wiki-based collaborative monitoring of IT standards, in partial fulfillment of the requirements of Task 3.1 in WP3. It is largely intended as a catalogue of potentially useful standards, and comes to no firm conclusions regarding which should be used under p-medicine, a task which will be carried out during architecture design and throughout the project when required.

Introduction

The introduction gives some loose definitions of what a standard is. In the work we have been quite liberal with the definition of a standard, including guidelines and software within our remit. The introduction also lists some of the feature of standards which might make them attractive, and features which might make them less useful.

Standards and guidelines from VPH toolkit

This section gives an overview of the eight Guideline documents which have be produced by the VPH NoE to set standards for tools to be submitted to the VPH toolkit portal. These guidelines range from toolkit, model and data characterisation to ontology, interoperability, licensing, usability as well as legal & ethics guidelines. Content deliverers can use these guidelines to assess what are the ideal content as well as how to present their content to the wider audience through the VPH-NoE toolkit portal. The VPH guidelines should be adhered to by tool authors wishing to distribute their tools though the VPH NoE. They are relevant to all work packages which produce IT solutions.

Software engineering standards

This section describes standards and guidelines relating to all aspects of software engineering:

- Requirements gathering, recording and expression
- Design, including design patterns and modelling methods such as UML
- Implementation methodologies, such as Agile and Waterfall, and basic code formatting and testing standards
- Sustainability issues, including evaluation, maintenance and licensing issues

- Quality assurance

Few formal standards are employed in software engineering, but many informal standards are followed by convention, and this section makes some suggestions as to which might be useful. The formal standards are predominantly defined by IEE, requiring payment, and are generally regarded as too verbose to justify following them.

Software engineering standards are relevant to all work packages producing IT solutions. Although the p-medicine architecture may be loosely coupled components, and little collaborative development or code sharing may take place, it is important we adhere to best practice in software engineering to ensure sustainability of p-medicine solutions.

Human-computer interface and web standards

Many work packages will produce software or servers with a web interface. This section details the standards relevant to this, including protocols such as HTTP, formats such as HTML and JPEG, client-side execution models such as JavaScript and Flash, and usability standards which ensure a high quality user experience. It is vitally important that p-medicine web interfaces adhere to these standards, partly for technical reasons - a web interface is not a web interface if it does not adhere to them - and partly to ensure the interfaces are accessible on a wide range of platforms.

Data exchange formats and protocols

This section describes standards for exchanging clinical data, imagery, and standards for output from and input to modelling and data mining tools. Standards in these areas range from very formal and defined by a standards body, such as CDISC and HL7 standards in clinical data, and very informal ad hoc poorly defined standards, such as input and output for bioinformatics tools. Use of standards in this domain must therefore be decided on a case by case basis.

Security standards

This section provides extensive information about the standards relating to IT security, including identity management, authorisation, data storage and data transfer. Clearly, given the sensitivity of the data stored in p-medicine, it is vital that security is given top priority. Adhering to standards allows interoperation with existing, thoroughly tested security solutions. However, the wide range of standards in this area makes choosing between them difficult - although multiple standards might be supported.

Interoperability standards

Many parts of p-medicine involve making tools and systems work together - for example, assembling tools into workflows, deploying them into cloud infrastructure, or supplying them with data from the data warehouse. This section describes standards relating to these tasks. A particular focus is upon semantic web standards, which provide a widely used framework for exchanging and searching structured data which can enhance interoperability.

Standards relating to specific work packages

Some work packages have specific sets of standards relating to them, which this section describes.

Data warehousing standards

The data warehouse (WP7) must make particular reference to standards for integrating data, accessing and querying data and storing data. Other than generic data storage and querying standards such as RDF and SPARQL, there are few useful standards relating specifically to data warehousing.

An important requirement in p-medicine is that the data in the warehouse is curated and audited, and there are more standards relating to this, particularly ontologies for annotation of data, formal standards for logging and audit trails, and ontologies and Minimum Information projects for provenance information. These standards are worth following in order to interoperate with relevant tools.

Biobanking standards

While various institutions have issued best-practices-guidelines for the management of biomaterial repositories, which cover only some of the above aspects on the biomaterial level and on the ethical and legal level, these standardisation aspects have not yet been sufficiently addressed for federated biobanks and respective meta biobank infrastructures. The state-of-the-art in federated biobanking is further investigated in WP10 and will be described in Deliverable 10.1.

Data mining standards

Generally speaking, in common with data warehousing, data mining lacks specific standards, with only a few exceptions. It does not rely on proprietary formats but uses standards developed for other purposes. This section describes, amongst other things, the widely-supported PMML for describing a predictive model, tools and APIs for use in data mining, and data exchange formats.

Contents

Glossary

ADaM *Analysis Data Model* CDISC standard supporting efficient generation, replication, and review of analysis results

AES *Advanced Encryption Standard* a specification for the encryption of electronic data based on a design principle known as a Substitution permutation network

AGPL *Affero General Public License* refers to two free software licenses. Affero General Public License, Version 1 and GNU Affero General Public License, version 3.

API *application programming interface* is a particular set of rules ('code') and specifications that software programs can follow to communicate with each other

ASCII *American Standard Code for Information Interchange* a character-encoding scheme based on the ordering of the English alphabet

BMP *Bitmap* a raster graphics image file format used to store bitmap digital images

BSD *Berkeley Software Distribution* is a Unix operating system derivative developed and distributed by the Computer Systems Research Group (CSRG) of the University of California, Berkeley

CAS *Central Authentication Service* a single sign-on web protocol

CA *Certification Authority* an entity that issues digital certificates

CBC *Cipher-Block Chaining* a cryptographic mode of operation in which each block of plaintext is XORed with the previous ciphertext block before being encrypted

CCM *Counter with CBC-MAC Mode* a mode of operation for cryptographic block ciphers

CCZero *Creative Commons licenses* are several copyright licenses that allow the distribution of copyrighted works

CDASH *Clinical Data Acquisition Standards Harmonization* CDISC standard describing the basic recommended (minimal) data collection fields for 18 domains, including common header fields, and demographic, adverse events, and other safety domains that are common to all therapeutic areas and phases of clinical research

CDA *Clinical Document Architecture* is an XML-based markup standard intended to specify the encoding, structure and semantics of clinical documents for exchange

CDE *Clinical Document Architecture* an XML-based markup standard defined by HL7 intended to specify the encoding, structure and semantics of clinical documents for exchange

CDISC *Clinical Data Interchange Standards Consortium* - a global, open, multidisciplinary, non-profit organization that has established standards to support the acquisition, exchange, submission and archive of clinical research data and metadata

CDMI *Cloud Data Management Interface* - defines a functional interface that applications can use to create, retrieve, update and delete data elements from the Cloud

CDS *Clinical Decision Support* decision support software designed to assist physicians and other health professionals with decision making tasks, as determining diagnosis of patient data

CMWG *Cloud Management Work Group* focused on standardizing interactions between cloud environments by developing specifications that deliver architectural semantics and implementation details to achieve interoperable cloud management between service providers and their consumers and developers

CRISP-DM *Cross Industry Standard Process for Data Mining* a data mining process model that describes commonly used approaches that expert data miners use to tackle problems

CRL *Certificate Revocation List* a list of certificates that have been revoked, and therefore should not be relied upon

CSS *Cascading Style Sheets* is a style sheet language used to describe the presentation semantics (the look and formatting) of a document written in a markup language

CSV *Comma-Separated Values* a set of file formats used to store tabular data in which numbers and text are stored in plain-text form that can be easily written and read in a text editor

CWM *Common Warehouse Metamodel* a specification for modeling metadata for relational, non-relational, multi-dimensional, and most other objects found in a data warehousing environment

CeCILL *CEA CNRS INRIA Logiciel Libre* is a free software license adapted to both international and French legal matters, in the spirit of and retaining compatibility with the GNU General Public License

CellML *Cell Markup Language* is an XML based markup language for describing mathematical models

DICOM *Digital Imaging and Communications in Medicine* - a standard for handling, storing, printing, and transmitting information in medical imaging

DTMF *Distributed Management Task Force* brings the IT industry together to collaborate on the development, validation and promotion of systems management standards

EHR *electronic health record* is an evolving concept defined as a systematic collection of electronic health information about individual patients or populations

EULA *end-user licensing agreements* An EULA is a legal contract between the manufacturer and/or the author and the end user of an application

EUPL *European Union Public Licence* the first European Free/Open Source Software (F/OSS) licence

FMA *Foundational Model of Anatomy* it is concerned with the representation of classes or types and relationships necessary for the symbolic representation of the phenotypic structure of the human body in a form that is understandable to humans and is also navigable, parseable and interpretable by machine-based systems

FieldML *Field Markup Language* is an XML based markup language for describing field models

GAS *Gridge Authorization Service* provides functionality that would be able to fulfill most authorization requirements of grid computing environments

GCM *Galois/Counter Mode* a mode of operation for symmetric key cryptographic block ciphers that has been widely adopted because of its efficiency and performance

GEM *Guideline Elements Model* an XML-based guideline document model that can store and organize the heterogeneous information contained in practice guidelines

GNU *Gnu's Not Unix* is a Unix-like computer operating system developed by the GNU project, ultimately aiming to be a "complete Unix-compatible software system" composed wholly of free software.

GO *Gene Ontology* is a major bioinformatics initiative with the aim of standardizing the representation of gene and gene product attributes across species and databases

GPL *General Public License* is the most widely used free software license, originally written by Richard Stallman for the GNU Project

GridFTP *GridFTP* is an extension of the standard File Transfer Protocol (FTP) for use with Grid computing

HL7 *Health Level Seven* is an all-volunteer, non-profit organization involved in development of international healthcare informatics interoperability standards

HMAC *Hash-based Message Authentication Code* a mechanism for message authentication using cryptographic hash functions

HTML *Hypertext Markup Language* is the predominant markup language for web pages. HTML elements are the basic building-blocks of webpages.

HTTPS *Hypertext Transfer Protocol Secure* is a combination of the Hypertext Transfer Protocol (HTTP) with SSL/TLS protocol to provide encrypted communication and secure identification of a network web server

IBM *International Business Machines*

ID-FF *Liberty Identity Federation Framework* an approach for implementing a single sign-on with federated identities based on commonly deployed technologies

ID-WSF *Liberty Identity Web Services Framework* a framework for identity-based web services in a federated network identity environment

IEC *International Electrotechnical Commission* is the worlds leading organization that prepares and publishes International Standards for all electrical, electronic and related technologies

IEEE *Institute of Electrical and Electronics Engineers* is a non-profit professional association headquartered in the United States that is dedicated to advancing technological innovation and excellence

IETF *Internet Engineering Task Force* a large open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet

IHE *Integrating the Healthcare Enterprise* - an initiative by healthcare professionals and industry to improve the way computer systems in healthcare share information

IPSec *Internet Protocol Security* a protocol suite for securing Internet Protocol (IP) communications by authenticating and encrypting each IP packet of a communication session

ISBN *International Standard Book Number* is a unique numeric commercial book identifier based upon the 9-digit Standard Book Numbering (SBN) code created by Gordon Foster

ISO *International Organization for Standardization* is an international standard-setting body composed of representatives from various national standards organizations

InSilicoML *InSilico Markup Language* is a markup language that can explicitly describe the multi-level hierarchical structures of the physiological functions in mathematical models

JDMP *Java Data Mining Package* an open source Java library for data analysis and machine learning

JPEG *Joint Photographic Experts Group* is a commonly used method of lossy compression for digital photography

JSDL *Job Submission Description Language* is an extensible XML specification from the Global Grid Forum for the description of simple tasks to non-interactive computer execution systems

KNIME *Konstanz Information Miner* a user-friendly and comprehensive open source data integration, process, analysis and exploration platform

LGPL *Lesser General Public License* is a free software license published by the Free Software Foundation

LOINC *Logical Observation Identifiers Names and Codes* is a database and universal standard for identifying medical laboratory observations

MAGE-ML *Microarray and Gene Expression - Markup Language* markup language format for the representation of gene expression data from microarrays to facilitate the exchange of information between different data systems

MAGE-OM *Microarray and Gene Expression - Object Model* data exchange model for the representation of gene expression data from microarrays to facilitate the exchange of information between different data systems

MAGE-TAB *Microarray and Gene Expression - Tabular* tabular format for the representation of gene expression data from microarrays to facilitate the exchange of information between different data systems

MIAME *Minimum Information About a Microarray Experiment* needed to enable the interpretation of the results of the experiment unambiguously and potentially to reproduce the experiment

MIASE *Minimal Information About a Simulation Experiment* common set of information a modeller needs to provide in order to enable the execution and reproduction of a numerical simulation experiment, derived from a given set of quantitative models

MIASE *Minimum Information About a Simulation Experiment* is an effort to list the common set of information a modeller needs to provide in order to enable the execution and reproduction of a numerical simulation experiment, derived from a given set of quantitative models.

MIBBI *Minimum Information for Biological and Biomedical Investigations* maintains a web-based, freely accessible resource for "Minimum Information" checklist projects, providing straightforward access to extant checklists (and to complementary data formats, controlled vocabularies, tools and databases), thereby enhancing both transparency and accessibility

MIT *MIT License* is a free software license originating at the Massachusetts Institute of Technology

ML *Markup Language* is a modern system for annotating a text in a way that is syntactically distinguishable from that text

MOF *MetaObject Facility* the foundation of OMG's industry-standard environment where models can be exported from one application, imported into another, transported across a network, stored in a repository and then retrieved, rendered into different formats

MPL *Mozilla Public License* is a free and open source software license

MS *Microsoft* is an American public multinational corporation headquartered in Redmond, Washington

MTOM *Message Transmission Optimization Mechanism* is the W3C Message Transmission Optimization Mechanism, a method of efficiently sending binary data to and from Web services

MedLEE *Medical Language Extraction and Encoding system* System to extract, structure, and encode clinical information in textual patient reports so that the data can be used by subsequent automated processes

NeuroML *Neuro Markup Language* is an XML (Extensible Markup Language) based model description language that aims to provide a common data format for defining and exchanging models in computational neuroscience

OASIS *Organization for the Advancement of Structured Information Standards* a not-for-profit consortium that drives the development, convergence and adoption of open standards for the global information society

OBO *Open Biomedical Ontologies* is an effort to create controlled vocabularies for shared use across different biological and medical domains

OGSA-BES *Open Grid Services Architecture - Basic Execution Services* defines Web Services interfaces for creating, monitoring, and controlling computational entities such as UNIX or Windows processes, Web Services, or parallel programswhat we call activities within a defined environment

OGSA-DAI *Open Grid Service Architecture-Data Access and Integration* allows data resources (e.g. relational or XML databases, files or web services) to be federated and accessed via web services on the web or within grids or clouds. Via these web services, data can be queried, updated, transformed and combined in various ways.

OSI *Open Source Initiative* is an organization dedicated to promoting open source software

OS *Operating System* is a set of programs that manages computer hardware resources, and provides common services for application software

OWL-S *Ontology Web Language for web Services* an ontology of services to discover, invoke, compose, and monitor Web resources offering particular services and having particular properties

OWL *Web Ontology Language* is a family of knowledge representation languages for authoring ontologies.

OpenID *Open Identity* provider of web-based SSO services

PAOS *Reverse HTTP Binding for SOAP* a binding that enables HTTP clients to expose services using the SOAP protocol, where a SOAP request is bound to a HTTP response and vice versa

PATO *PATO* an ontology of phenotypic qualities, intended for use in a number of applications, primarily defining composite phenotypes and phenotype annotation.

PHP *PHP: Hypertext Preprocessor* is a general-purpose server-side scripting language originally designed for web development to produce dynamic web pages

PKIX *Public-Key Infrastructure* Working Group was established in the fall of 1995 with the goal of developing Internet standards to support X.509-based Public Key Infrastructures

PMML *Predictive Model Markup Language* an XML-based language which provides a way for applications to define statistical and data mining models and to share models between PMML compliant applications

PNG *Portable Network Graphics* is a bitmapped image format that employs lossless data compression

POST *POST* is one of many request methods supported by the HTTP protocol used by the World Wide Web

RAD *Rapid application development* is a software development methodology that uses minimal planning in favor of rapid prototyping

RDF *Resource Description Framework* is a family of World Wide Web Consortium (W3C) specifications originally designed as a metadata data model

REST *Representational state transfer* is a style of software architecture for distributed hypermedia systems such as the World Wide Web

RFC *Request for Comments* is a memorandum published by the Internet Engineering Task Force (IETF) describing methods, behaviors, research, or innovations applicable to the working of the Internet and Internet-connected systems

RICORDO *RICORDO* is focused on the study and design of a multiscale ontological framework in support of the Virtual Physiological Human community to improve the interoperability amongst its Data and Modelling resources

RIM *Reference Information Model* is the cornerstone of the HL7 Version 3 development process and an essential part of the HL7 V3 development methodology

SAML *Security Assertion Markup Language* a standard, XML-based framework for creating and exchanging security information between online partners

SAS business analytics software and service developer, and independent vendor in the business intelligence market

SAWSDL *Semantic Annotations for WSDL* defines mechanisms using which semantic annotations can be added to WSDL components

SBML *System Biology Markup Language* is a representation format, based on XML, for communicating and storing computational models of biological processes

SDTM *Study Data Tabulation Model* CDISC defining a standard structure for human clinical trial (study) data tabulations that are to be submitted as part of a product application to a regulatory authority

SED-ML *Simulation Experiment Description Markup Language* an XML-based format for encoding simulation experiments, following the requirements defined in the MIASE guidelines

SHA *Secure Hash Algorithm* a number of cryptographic hash functions published by the National Institute of Standards and Technology as a U.S. Federal Information Processing Standard

SLO *Single Log-Out* termination of a SSO action

SNIA *Storage Networking Industry Association* not-for-profit trade organization for companies and individuals in various sectors of the storage industry

SNOMED-CT *Systematized Nomenclature of Medicine - Clinical Term* is a systematically organised computer processable collection of medical terminology covering most areas of clinical information such as diseases, findings, procedures, microorganisms, pharmaceuticals etc

SOAP *Simple Object Access Protocol* is a protocol specification for exchanging structured information in the implementation of Web Services in computer networks

SOAP *Simple Object Access Protocol* a lightweight XML-based protocol for exchange of structured information in a decentralized, distributed environment

SOA *Service-Oriented Architecture* s a set of principles and methodologies for designing and developing software in the form of interoperable services

SPARQL *SPARQL Protocol and RDF Query Language* - query language for RDF

SQL *Structured Query Language* a standard language for accessing and manipulating databases

SSH *Secure Shell* is a network protocol for secure data communication, remote shell services or command execution and other secure network services between two networked computers that it connects via a secure channel over an insecure network

SSL *Secure Sockets Layer* a cryptographic protocol that provides communication security over the Internet, predecessor of TLS

SSO *Single Sign-On* a mechanism whereby a single action of user authentication and authorization can permit a user to access all computers and systems where he has access permission, without the need to enter multiple passwords

TCP/IP *Transmission Control Protocol/Internet Protocol* the first two networking protocols defined in the Internet Protocol Suite standard

TDD *Test-driven development* is a software development process that relies on the repetition of a very short development cycle.

TLS *Transport Layer Security* a cryptographic protocol that provides communication security over the Internet, successor of SSL

UML *Unified Modelling Language* a specification defining a graphical language for visualizing, specifying, constructing, and documenting the artifacts of distributed object systems

VDM *Vienna Development Method* is one of the longest-established Formal Methods for the development of computer-based systems

VPH-NoE *Virtual Physiological Human - Network of Excellence* is a project which aims to help support and progress European research in biomedical modelling and simulation of the human body

VPH *Virtual Physiological Human* is a methodological and technological framework that, once established, will enable collaborative investigation of the human body as a single complex system

WAV *Waveform Audio File Format* is a Microsoft and IBM audio file format standard for storing an audio bitstream on PCs

WS-* *Web Services-** common prefix for the family of Web Services specifications

WSDL *Web Services Description Language* a way to describe the abstract functionalities of a service and concretely how and where to invoke it

WSMO *Web Service Modelling Ontology* ontology for describing Semantic Web Services

XFree86 a freely redistributable open-source implementation of the X Window System

XHTML *eXtensible HyperText Markup Language* is a family of XML markup languages that mirror or extend versions of the widely-used Hypertext Markup Language (HTML), the language in which web pages are written

XML *Extensible Markup Language* - a format for encoding documents in machine-readable form, similar in syntax to HTML

XTS *XEX-based Tweaked Codebook* a mode of operation for cryptographic block ciphers

caBIG *cancer Biomedical Informatics Grid* a virtual network of interconnected data, individuals, and organizations that work together to redefine how cancer research is conducted

1 Introduction

Relevant to all work packages

1.1 What is a standard?

The range of standards can be divided in different ways. They can be divided into two groups according to the way they came about (text from Wikipedia):

- *Technical standard:* "a formal document that establishes uniform engineering or technical criteria, methods, processes and practices"
- *De-facto standard:* "a custom, convention, product, or system that has achieved a dominant position by public acceptance or market forces"

HTML is a good example of both: it started out as a de-facto standard and was transformed into a technical standard. Standards might also be divided broadly by what they govern - this is taken directly from Wikipedia:

- A *standard specification* is an explicit set of requirements for an item, material, component, system or service. It is often used to formalize the technical aspects of a procurement agreement or contract. For example, there may be a specification for a turbine blade for a jet engine which defines the exact material and performance requirements.
- A *standard test method* describes a definitive procedure which produces a test result. It may involve making a careful personal observation or conducting a highly technical measurement. For example, a physical property of a material is often affected by the precise method of testing: any reference to the property should therefore reference the test method used.
- A *standard practice* or procedure gives a set of instructions for performing operations or functions. For example, there are detailed standard operating procedures for operation of a nuclear power plant.
- A *standard guide* is general information or options which do not require a specific course of action.
- A *standard definition* is formally established terminology.
- *Standard units of measurement*, in physics and applied mathematics, are commonly accepted measurements of physical quantities.

We will primarily be concerned with standard specifications, practices and guides. Less formally, the following entities can be considered as standards:

- Software
- Guidelines
- Protocols
- Formats
- Laws

We are primarily concerned here with IT standards. However, standards that do not specifically refer to IT architecture might *influence* IT architecture, and must be considered (perhaps briefly).

1.2 Motivation for using standards

There are several advantages of making use of a standard:

- It **enables interoperability** with other software which uses the same standard
  - Related: It makes a software system potentially a **drop-in replacement** for another system, encouraging adoption
- It **avoids doing work** developing new protocols, formats, software and other entities which are typically associated with a standard
- It makes software systems easier to adopt, maintain and develop by new people who are familiar with the standard
- It may be a **legal requirement** (or the standard may take the form of a legal requirement)
- Some standards enforce minimum **quality standards**
  - Related: It **lends credibility** to a software system, making it more likely to be adopted, particularly in mission critical or life-or-death applications

There are also some disadvantages of making use of a standard:

- It is **not widely adopted**, nullifying many of the above advantages
- It may be **complex or heavyweight** and therefore hard to comply to
- It may be so **lightweight** that few benefits are gained from adhering to it, compared to the effort put in
- It might **not be a good fit** for the software system being developed - it may lack required elements, or make demands unnecessary for the task at hand

- It is in development or **constantly changing**, making compliance difficult
- There are many **competing standards** and there is no clear picture of which will ultimately win

2 Standards and guidelines from VPH toolkit

Relevant to WP4, WP7, WP8, WP10, WP11, WP12, WP13, WP14, WP15, WP16, WP17

2.1 Introduction

The VPH initiative is expected to deliver a wide range of content, including tools, data and models. To assist in the dissemination of this content through a central point of access, the **VPH network of excellence (VPH-NoE)** project is developing a series of guidelines [1] that will address relevant issues to content deliverers who wants to share their content through the VPH-NoE toolkit portal. These guidelines range from toolkit, model and data characterisation to ontology, interoperability, licensing, usability as well as legal & ethics guidelines. Content deliverers can use these guidelines to assess what are the ideal content as well as how to present their content to the wider audience through the VPH-NoE toolkit portal. The p-medicine project is also pursuing a similar strategy with a distributed portal to share information from data to software. The VPH guidelines provide a useful set of information, a starting point on how to classify the content, and ancillary information related to the content itself. A brief overview of the key aspects of the individual guidelines will be provided in the following sections.

2.2 Toolkit characterisation

The purpose of the toolkit characterisation guidelines [2] is to present a set of criteria that will identify the functionalities of the toolkit in its domain as well as a method to assess the quality of the toolkit. This guideline also aims to provide a standard on what toolkits should be accepted to the VPH-NoE portal.

The toolkit guideline identifies a number of areas of standards that are applicable to toolkits including input/output format, language, operating system, third party library, documentation, maintenance & versioning, license and certification. The adoption of standards by the toolkit is considered as a characteristic of the toolkit. Other areas of characterisation includes:

- Tool information such as name, version, format and function
- Tool specification which includes language, operation system, installation recommendation and tool description

- Tool descriptions such as description of purpose, documentation link, keywords, testing and citation & references
- Tool context including authorship, support, type of collaboration, funding status and targeted users
- Tool functionality and speciality which uses several ontologies that can be used to characterised the toolkit. These includes, Biomeidcal Resource Ontology, BIRNLEX, NEUROLEX, Software Ontology, SW Tool Ontology and OntoNeuro
- Tool usability

This guideline also discusses the method of verification, ownership, training, maintenance as well as suggested method of ranking. Although this guideline is not complete, it contains much useful information and a starting point on the approach and methods to classify toolkits for the p-medicine project.

2.3 Model characterisation

The goal of the model characterisation guideline [3] is to ensure that models can be understood by the end user, coded and solved. This has become increasingly important as models have become more complex. Furthermore, to be used in a clinical environment, a model must be able to be validated and demonstrated to be reproducible. This guideline focuses on issues which address these problems specifically, the development of specification of the minimum information required to describe a model as well as the development of model encoding standards. The minimum information specification described in this guideline includes the MIRIAM (Minimal Information Required in Annotation of biochemical model), MIASE (Minimum information about a simulation experiment) and MIBBI (Minimum information about a biomedical or biological investigation). Other "Minimum information" projects are described in **Data exchange formats and protocols** and **Data warehousing** sections.

Several Encoding standards have also been mentioned in this guideline including CellML, FieldML, SBML as well as NeruoML, InSilicoML. The typical strategy for developing an encoding standards includes:

- Development of a markup language including metadata and data
- Development of an application programming interface based on the MLs
- Development of libraries of tools that can read and write ML files
- Development of data and model repository based on MLs
- Development of metadata framework that demonstrates model reproducibility

This guideline also describes validation methods, training, maintenance, documentations as well as method of ranking. Not all sections are completed. However this guideline provides a general overview on common used standards and specifications in the area of modelling that could be adopted by p-medicine.

2.4 Data characterisation

The exchange of data between different users is an important goal of the VPH toolkit portal. The goal of this guideline [4] is to facilitate the exchange of the data through a trusted mechanism. This guideline first defines the scope of data characterisation principles including: ethics, law, licensing, reproducibility, interoperability and sustainability. It also discusses data sharing based on quality standards and good practice such as technical characterisation, provenance and standard formats. Other areas of this guideline includes, existing data-sharing plans as well as possible future work in this guideline.

The Data guideline recommends data licenses which are compliant with the The Open Knowledge definition developed by the Open Knowledge foundation. The suitability of the licenses are assessed based on the following criteria

- Whether or not it conforms to the Open Knowledge definition
- Whether it allows commercial use
- Whether is a copyleft licence
- Whether you need to acknowledge the work that the data is derived from
- Legal jurisdiction

Ethics and data are described in a number of areas including the ethical use of data within the VPH project, personal data as well as health data. The importance of ethics within the VPH project is considered to be an important factor on the longevity of the project, as such the use of material within the VPH should acknowledge the right of those that contributed the data including specific right to publish, the role of all contributors as well as compliance with relevant legal requirements. The **European Data protection directive (Directive 95/46/EC)** is designed to protect the privacy of personal data and with freedom of information measure (**European Union Directive 2003/98/EC**) supports the right of individual to inspect the nature of information held about them. Ethics in terms of health data requires that contributors can be expected to:

- Withdraw or refuse the use of their data at any time
- Their data is treated with respect
- Their data is treated confidentially

- Exploitation of their data will not expose them to unnecessary level of risk

These principles can be found at the **Research Ethics, Committees, Data Protection and Medical Research in European Countries, Directive 2001/20/EC**.

The data guideline characterises data according standards used as well as technical aspects such as the inclusion of metadata which complements the raw data; data processing which describes how the data may have been processed to allow repetition of a study; interoperability which describes how easy is it to read or write the data including documentation and openness and sustainability which related to the data format used and how it will be stored. The criteria of sustainability for file formats includes:

- Extensibility
- User base
- Licensing
- Backward compatibility
- Stability

The data characterisation guidelines also provide use case data sharing plans from biosharing, cardiac atlas project, and medical research council data sharing initiative as well as the Open Provenance Model. This guideline provides a number of important aspects on how data can be treated and what ethical guidelines that p-medicine should be aware of.

2.5 Ontology guidelines

The ontological guidelines [5] are intended to support the verifiability and re-use of data and model resources. A key aspect of this is to develop a strategy for data and model resources that is both human and machine readable. These include metadata to hold resource annotation and use of knowledge representation and ontologies in annotation. Within the VPH project, the key approach is to develop and support semantic interoperability between data and model resources. The semantic interoperability aim to provide a standardized as well as machine readable context to data. This will allow interpretation of different resources more easily. This guideline uses a dedicated semantic integration framework developed through the RICORDO VPH project [6]. This guideline provides a useful overview of the scope of ontology, semantic integration and the relevance of these to the VPH project. This guideline makes the following recommendation for the following areas:

- Biological structures
  - The foundational model of anatomy

  - Edinburgh Mouse Atlas (EMAP)
  - Mouse Anatomical (MA) dictionary
  - The Cell Type ontology
  - Gene ontology cell component
  - Protein ontology
  - Chemical Entities of Biological Interest
- Biological Events
  - Go Biological Process
  - Mammalian Pathology
- Qualities
  - Go Molecular Function
  - PATO
  - Ontology for Physics in Biology
- Classes of entities in Experiments, Modeling and Simulation
  - Units Ontology
  - Ontology for Biomedical Investigation

The RICORDO project attempts to develop a grammar that draws references from basic ontologies to create a composite representation of complex biological concepts. The onotology guideline also list a range of knowledge representation tools relevant to the VPH community that could also be used in the p-medicine project. These include:

- Knowledge Representation Languages
  - Prolog-based: found in SWI-Prolog system
  - LISP-based: OCML
  - CommonLogic
  - OBO format
- Web Languages
  - RDF
  - RDF-Schema
  - OWL

The ontology guideline also briefly mention tools in authoring, browsing & validation, reasoning and management as well as methods of deployment. No specific mentions of tools or applications are mentioned here. The ontology guideline acknowledges the difficulty in assessing and ranking the usefulness of any ontology. However, it mentions some possible criteria to assess ontologies including:

- Absolute generic criteria
  - Availability
  - Usability
  - Documentation and training
  - Robustness of validation procedures
- Context dependent criteria
  - Integration and interoperability
  - Adoption of communal standards
  - Strength and commitment of developer community and users
  - Quality
  - Generic application

2.6 Interoperability Guidelines

Interoperability according to International Standards and Organization (ISO/IEC 2382-01, Information Technology Vocabulary and Fundamental Terms) can be defined as "the capability to communicate, execute programs, or transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristic of those units". The interoperability guideline [7] attempts to provide a set of practical interoperability recommendation in specific domains. Interoperability is seen as an important aspect in the effort to integrate tools, data and model to support the development of patient specific computer models and their applications in the VPH projects. According to the Interoperability guideline, interoperability can be classified in the following manner:

- Level 0: Standalone system with no interoperability
- Level 1: Technical interoperability with communication protocol available for exchanging data.
- Level 2: Syntactic Interoperability where a common structure is available to exchange information. i.e. common data format.
- Level 3: Semantic Interoperability where a common information exchange model is used i.e. ontology.

- Level 4: Pragmatic Interoperability where each system is aware of the methods and procedure of the other systems.
- Level 5: Dynamic Interoperability where a system can comprehend the state changes that occur in the assumptions and constraints each is making over time.
- Level 6: Conceptual Interoperability where the system is fully specified and be able to be evaluated by other engineers.

The interoperability guidelines list out the following standards or commonly used tool in the following domains:

- Modelling: CellML, FieldML, SBML, NeuroML
- Data: DICOM, HL7, JPEG, TIFF, VTK file format, BioSignalML, GDF, Analyse 7.5, Nifti
- Ontology: FMA, SNOMED CT, GO, LOINC, MIASE
- Infrastructure: JSDL, OGSA-BES, OGSA-DAI, GridFTP, SSH, MTOM

Interoperability of models, data, infrastructure and tools can be characterised. The guidelines outline the need to deploy common standards as well as issues related to each specific area including legacy systems. The guideline suggests the use of IHE (Integrating the Healthcare Enterprise) as a means of verification. IHE Gazelle Tools has been developed to test interoperability according to the IHE standards.

2.7 Legal, ethics and provenance

The Legal, Ethical & Provenance guidelines [8] describes the legal and ethical requirements when interacting with the VPH toolkits, including copyright, data protection and freedom of information. The close association of VPH with industry and clinic requires both provenance and ethical standards to be followed.

The guideline raises practical legal issues relevant to the delivery of content to the toolkit. It is categorised into sections on simulation & data, security, research & innovation, tools & techniques, interoperability & workflow and standards. The guidelines point out that the common legal issues which arise regularly includes whether licensing has been clearly stated and attached, whether all involved parties have been acknowledged and the copyright conditions clearly stated and the whether an adequate disclaimer have been attached to the toolkit content. These legal issues are relevant to all users. Personal interaction with the toolkit will result the user being classified either as a user or author and hence certain legal responsibility will fall on the user.

The ethical guidelines highlight the issues relevant to VPH toolkit users which can then be used as a framework on the definition of *acceptability* and emphquality. Relevant ethical issues are

categorised into sections on software & data, community, sustainability of infrastructure, exploitation and sustainability of interoperability and workflow. The ethical guidelines can be considered as a best practice guide. Given the close association of VPH with industry and clinicians, the need to follow ethical standards closely are emphasised. A basic principle of consent, authorisation and anonymization should be adopted.

The guidelines also a brief background on the significance and importance of provenance. The use of **Open Provenance Model** is recommended. The issues of provenance is discussed in the following categories: protocol, quality & provenance in data and software, security, simulation, provenance protocol, research and innovation, tools and techniques, interoperability and workflows, sustainability in community, ontology, documentation, exploitation.

The legal and ethical guidelines provides a number of use case studies. Given the similarities and overlapping of some VPH projects and p-medicine, the issues and standards raised in these guidelines are particularly relevant to the scope of p-medicine given the close association to clinical data and software development.

2.8 Licensing

The licensing guidelines [9] highlight the issues involved with software, data and content licensing. This guidelines recommend using business-friendly open source licence whenever possible as well as relevant open license for data if applicable. This guideline provides a background and underlying concept of licensing as well as standards and standard bodies such as:

- Open Source Initiative (software licensing)
- Free Software Foundation
- Open Knowledge Foundation
- Creative Commons
- Science Commons

The guidelines recommend specific licenses, with a description of relevant characteristics. Software, data and content licenses are characterised according to 5 criteria, including OSI approval, business-friendliness, GNU GPL (viral copyleft license) compatiblity, attribution and the legal jurisdiction specified in the license.

- Software licenses
  - AGPL - The GNU Affero General Public License v3
  - Apache v2 - The Apache Licence

  - 3-clause BSD - The 3-clause BSD Licence
  - 4-clause BSD - The 4-clause BSD Licence
  - CeCILL v2 - The CeCILL Licence
  - EUPL - The European Union Public Licence
  - GPL v2 - The GNU General Public Licence version 2
  - GPL v3 - The GNU General Public Licence version 3
  - LGPL v2.1 - The GNU Lesser General Public Licence version 2.1
  - MIT - The MIT Licence
  - MPL v1.1 - The Mozilla Public Licence 1.1
- Data licenses
  - Open Data Commons public domain dedication and licences
  - Open Data Commons attribution license
  - Open Data Commons Open Database Licenses
  - Creative Commons CCZero
- Content Licences
  - Creative Commons Attribution licence family
  - Creative Commons CCZero
  - GNU Free Documentation License

The VPH Toolkit welcomes all kinds of contents release in any licences. However, it recommends users to adopt OSI-approved business-friendly licenses to encourage the growth of the toolkit. The licensing guideline is particularly useful to the p-medicine project due to the development environment as well as the amount of data involved. It should be noted that due to the clinical nature of p-medicine project, legal and ethical issues must also be considered when choosing a license.

2.9 Usability

The usability guidelines [10] attempt to provide end-users with information on how to evaluate and assess the usability of the tools in the VPH toolkit. The usability criteria are defined as followed:

- Tool portability
- Installing the tool
- Defining the benchmark tasks

- Participant numbers
- Documentation
- Interoperability
- Learnability

Usability is an important issue on the uptake and acceptable of tools. A carefully thought-out design process with the use of a formal usability evaluation procedure is recommended to ensure a level of acceptable usability. The following usability design principle relevant to VPH includes:

- Learnability: Who are the users, what do they know, and what can they learn? How easy is it for them to accomplish a basic task for the first time?
- Efficiency: How easy is it for a regular user to perform a task?
- Memorability: How easy is it to maintain proficiency?
- Errors: How are user errors handled and how easily can a user recover from these errors?
- Satisfaction: Are users happy with the tools?
- Can a user easily accomplish their task?

The usability guidelines provide a overview on what criteria should be assessed for the VPH toolkit portal as well as how usability can be assessed by content developers to improve their tool.

3 Software engineering standards

Relevant to WP2, WP3, WP4, WP7, WP8, WP10, WP11, WP12, WP13, WP14

The various steps and processes in software engineering are typically categorized IEEE Guide to the Software Engineering Body of Knowledge (SWEBOK), ISBN 0769523307 [11] as

- Analysis / Specification
- Architecture / Design
- Implementation / Testing
- Evaluation / Maintenance / Documentation

In the sections below we present the existing standards for each category as well as selected *de facto* standards/technologies in the software engineering industry. The p-medicine project is a large consortium of partners with backgrounds ranging from clinical to academic to industry. With such a wide ranging backgrounds, the sections below could be used as a reference or guide as possible software engineering models to adopt for different institutions.

3.1 Analysis

The Analysis and Specification process in software engineering usually comprises of the elicitation, analysis, specification and validation of the requirements for software. The software requirements are properties which must be exhibited by the software to solve a particular problem. An essential property of the software requirements, although difficult to achieve, is that they be verifiable. In order to achieve this, requirements should be expressed as clearly as possible, and where appropriate with quantity metrics and measurable properties. There are various methodologies to gather and document the software requirements, there are also defined standards for the requirements specification and there are also a great number of tools for the requirements management, tracking, verification and documentation, which we outline below.

3.1.1 Requirements gathering methodologies

To the best of our knowledge, there is not a standard or *de facto* standard on how to collect the requirements of a system. Although there exist guides and standards on how to record and express the requirements, there is not a standard process on how to acquire them. However, there are various proposed techniques for the elicitation process described in the references under *Further reading*.

The first and most important step in the requirements elicitation process is to correctly identify the stakeholders of the software under development. Different groups of stakeholders, such as end users, customers, software engineers, government regulators, might have different, or even conflicting, requirements. If we fail to identify all the correct stakeholders in the requirements elicitation process, we usually end up delivering a product which has not taken in to account the correct set of requirements, thus we build a wrong or problematic piece of software.

Apart from the stakeholders, there are also other sources from where the software architect or the requirements engineer can draw the requirements, such as the specific domain knowledge he might have or acquire from an advisor with expertise on the subject, the operational and organizational environment of the software and others. There are various methodologies which we use for gathering the software requirements, each one having its advantages or specific cases in which may be more appropriate.

Interviews Interviews are the classic way of asking users what they need from the end product, how they imagine it and what particular preferences they have from it. Follow up interviews are usually needed in order to progressively end up with clearly expressed requirements.

Questionnaires By using questionnaires, we can gather requirements from large user groups, since they can be distributed to many users simultaneously and processed automatically, e.g. online questionnaires. We can also give specific directions to the users through the questions and acquire more meaningful answers than the vague requirements which an interview might come up with, but there is also the danger to have biased questions and

thus biased answers, so designing a questionnaire demands good knowledge of the problem domain.

Prototyping Prototypes are a valuable tool to provide a fast insight of the users to the end product and understand better what kind of input information they need to provide. Paper or screen mock ups are an especially useful type of prototype when building user interfaces.

Brainstorming We can use brainstorming for acquiring requirements, especially when we have a good knowledge of the domain or when we can consult domain experts on it. Brainstorming can also help when we consult in parallel different stakeholder groups, although it needs a good organization and agenda of the discussion, else the conversations might end up with vague views and no clear requirements.

Scenarios/use cases By using use cases, we provide a framework to identify the usual operational and functional requirements of software without explicitly asking the users for the requirements but rather for their usual tasks, work routine and workflows. We can also introduce hypothetical scenarios of the kind "what if" and design the software accordingly.

Modelling By modeling a specific problem and splitting it up into various subcategories, tasks, stakeholders we can gather the requirements needed to solve the problem. This usually might imply that we also need to sketch or apply existing models about the domain, the usage, the user interface etc. Conceptual modeling usually helps in understanding the problem, not designing the solution. Modeling also allows for a more formal definition both of the problem and of the requirements, which allows also for a more systematic tracking and verification of the requirements.

Further reading

- J. Goguen and C. Linde, *Techniques for Requirements Elicitation* presented at International Symposium on Requirements Engineering, 1993
- G. Kotonya and I. Sommerville, Requirements Engineering: Processes and Techniques, John Wiley & Sons, 2000
- S. Robertson and J. Robertson, Mastering the Requirements Process, Addison-Wesley, 1999
- I. Sommerville and P. Sawyer, Requirements Engineering: A Good Practice Guide, John Wiley & Sons, 1997
- R.R. Young, Effective Requirements Practices, Addison-Wesley, 2001
- R.H. Thayer and M. Dorfman, eds., Software Requirements Engineering, IEEE Computer Society Press, 1997
- A Guide to the Business Analysis Body of Knowledge (BAKoK), ISBN 9780981129211

3.1.2 Recording and expressing requirements

The result of the requirements elicitation phase is usually a document with a log of the various requirements. When we refer to a domain or problem which involves both software and non-software components we end up with a System Requirements Specification document, while when we refer to a list of requirements specifically for software we call it Software Requirements Specification document. When dealing with complex systems, a separate document might be compiled especially for the recording of the high level system requirements from the domain perspective, which contains the conceptual modeling of the domain, usage scenarios as well as exemplar data and workflows. Such a document is called System Definition document. For both categories there are standards on how to record and express the concepts and the requirements, such as the following:

- IEEE standard 1362-1998, *IEEE Guide for Information Technology - System Definition - Concept of Operations (ConOps) Document*
- IEEE standard 1233-1996 *IEEE Guide for Developing System Requirements Specifications*
- IEEE standard 830-1998 *IEEE Recommended Practice for Software Requirements Specifications*
- User Requirements Notation (URN) [12]
- Goal-oriented Requirement Language (GRL) [13]

List of Requirements Management (RM) tools The requirements of a typical system usually are numerous. Requirements also have several properties, such as quantification of desired attributes, preconditions and postconditions, and other defining elements. Consequently, it is difficult to record, keep track, verify, refine, and manage in any other way all the requirements of a system, unless an automated Requirements Management tool is used to help in this task. Numerous such tools exist, both proprietary or provided as free software. Each of these tools might have also advanced capabilities such as testing the software against the defined requirements, or be part of more general set of tools for project management, software design and so on. In Appendix A we present a non-exhaustive list of Requirements Management tools. Compiling a comparison or a list of all their characteristics would be outside the scope of this document, especially since many of them are commercial products which are not available for free.

3.2 Design

Based on the requirements of all the stakeholders, we proceed to the design of the system, which can be split up to two steps; the high level Architectural design of the main elements and components of the system and the low level Detailed design which defines the specific behavior, interfaces and algorithms of the architectural components. In the design phase, we usually take into consideration overall properties of the software such as

- Interoperability and compatibility
- Robustness and fault tolerance
- Security
- Extensibility and reusability
- Modularity and maintainability These considerations may be designed and then implemented based on standards, with custom solutions or may not be implemented at all depending on the requirements of the software, their prioritization, the budget and the timeplan of the software. The design of a system, similarly to its specification through requirements, can be described by using standard documentation, such as:
- IEEE standard 1471-2000 - "IEEE Recommended Practice for Architectural Description for Software-Intensive Systems"
- IEEE standard 1016-1998 - "IEEE Recommended Practice for Software Design Descriptions"

3.2.1 Design patterns

Both in the architectural and the low level design of the system, we can identify some common techniques or patterns which can be applied to ease the design process. Some common techniques which are used in the process of designing a system are: abstraction, decomposition and modularization, encapsulation and information hiding, separation of interface and implementation and others. By using these techniques many systems end up using common design patterns and since these patterns have been used widely in software engineering for many years, there are numerous studies regarding the categorization of these patterns, their common attributes, the usual cases where these patterns apply and guidelines about their implementation strategy. Some indicative resources on the literature about design patterns, both in architectural and implementation layer include:

- Erich Gamma et al, *Design Patterns: Elements of Reusable Object-Oriented Software*, ISBN 0-201-63361-2, 1994
- SOA Patterns [14] - A Community site for Service Oriented Architecture (SOA) Design Patterns
- Patterns of Enterprise Application Architecture [15], ISBN 0321127420
- Wikipedia article about Design Patterns [16]
- Article about Design Patterns and their categorization [17]

3.2.2 Modelling standards

A common way to understand, clarify and communicate ideas about the interactions, the dependencies and the interfaces between the various modules of a system, or between the system itself with external systems, is by using models. Models can be represented by graphical or textual notations, forming a formal (visual or textual) language for describing a system. The most common example of a standardized general modeling language is UML; however, many model languages exist for specialized topics, such as BPML for business process modeling or WSML for web service modeling. A software design approach which relies in modeling is the Model Driven Architecture (MDA), which was proposed and standardized by the Object Management Group (OMG). In MDA, the definition of a system is done using a platform-independent model (PIM) by utilizing a domain-specific language (DSL). Then, given a platform definition model (PDM) which corresponds to a specific development environment, such as CORBA, .NET, Web services etc., the PIM can be translated to a platform specific model (PSM). This translation is usually performed by automated tools, which execute the corresponding mappings between the various models and the actual implementation mechanisms.

- ISO/IEC 24744 [18] - ISO standard for software engineering meta modelling for development methodologies
- Unified Modeling Language (UML) [19] - OMG Standard
- Model Driven Architecture (MDA) [20] - OMG Standard

3.3 Implementation

The software construction, or implementation, is the detailed creation of working and meaningful, based on its requirements, software through a combination of coding, verification, testing, integration and debugging. Through construction, software engineers constantly test their product against errors and defects, so the software implementation is closely linked to the testing procedure, although it might be necessary in many systems to have a follow up testing phase in order to ensure and guarantee some quality attributes or test the overall integration status of the system.

3.3.1 Development Methodologies

A common consideration in the software implementation is that, more often than not, the software requirements might change. Consequently, the software construction methodology should take this into account and be flexible enough to adapt to changing requirements, to be able to adopt new requirements added from the testing and evaluation phase and to verify that the end product is in accordance with its requirements. For that reason various methodologies have been developed over time, which set some guidelines on how to implement software. Below we

outline the models of some of the most known techniques, although many other variations of them exist - see *Further reading*

Waterfall Sequential design process. The Waterfall methodology is a sequential development approach in which development flows directly from one phase to the other (analysis, design, implementation, maintenance). Emphasis is given to strict planning, time schedule, milestones and deadlines of each phase, thus not allowing backtracking or flexibility on the development scheduling.

Iterative Cyclic process of prototyping, testing and refining. The Iterative methodology is a cyclic process as a response to the weaknesses of the (sequential) Waterfall model. It starts with an initial planning which is refined through iterations of waterfall-like procedures of analysis, design, implementation and evaluation. In these iterations, each phase builds on the previous ones by taking into consideration the difficulties, problems and modifications which have been introduced since the previous planning.

Spiral Design and prototype in stages The Spiral model combines elements of design and prototyping in stages. Starting from a minimal project, the software development progresses into setting more requirements, adding stakeholders, refining the implementation and adapting the design in order to gradually expand into a fully defined and implemented system.

Agile Changing requirements The Agile methodology is a group of methodologies, such as Scrum, Extreme Programming, Adaptive Software Development, Feature Driven Development and others, which share some common characteristics. The core characteristics of the Agile model, are the continuous adaptation of the software development based on a set of changing requirements, the rapid delivery of software by frequent releases of new working versions and the close cooperation of the end-users with the developers.

RAD - Rapid prototyping Rapid Application Development is a methodology which facilitates the software construction through iterations and refinements of prototypes. Starting from GUI (graphical user interface) mock ups or prototypes with reduced functionality, this methodology prioritizes the business needs and the requirements and proceeds to next versions of prototypes with added functionality. These throwaway prototypes usually have little or no documentation and have active user involvement and continuous evaluation in order to achieve fast delivery with low cost.

TDD (test driven development) ***Fake it till you make it*** The Test Driven Development is a development model which relies on late implementation of functionality, until it is actually needed in the system. It starts with prototype implementations, which are gradually developed based on failing unit tests or integration tests, which introduce newly needed functionality. It can be considered as a variation of the RAD, although it usually does not deliver working prototypes but it builds over the failing ones. Although it is accredited for development speed, flexibility and extensibility, it is also criticized for the lack of clear management strategy.

Cleanroom Based on formal methods, certifiable reliability and adherence to specifications The Cleanroom methodology is intended to produce software with a certifiable level of reliability and validation against its specification. The main principles of the Cleanroom methodology is that it uses formal methods for the specification and designing of a system, it uses automated tools or standard based quality control, and testing with statistical models which can statistically verify the reliability of the software and the estimated confidence which is implied. The next section outlines some examples of formal methods.

Formal Development Methodologies The formal methods for specification, development and verification of computer and software systems are driven by the expectation that mathematical analysis and formalism can contribute to the reliability and robustness of a system. However, the high cost of using formal methods to systems development, usually restricts their usage only to high integrity systems where safety, security or robustness is of utmost importance.

Petri Nets A Petri net is one kind of mathematical modeling language for the description of systems. A Petri net is a graph, in which the nodes represent transitions (i.e. possible events) and places (i.e. possible conditions). Petri nets, similarly to other modeling languages such as UML, can be visually represented with graphical notation, but unlike these languages the graphical notation has an exact mathematical definition of its semantics and its process theory, thus Petri nets provide a combination of visual modeling power and mathematical analyzability.

Z Notation The Z Notation is a formal specification language which is used for describing and modeling computing systems. It is based on standard mathematical notation and contains a standardized catalog of commonly used mathematical functions and predicates. Since 2002 it has been standardized, under the ISO/IEC 13568:2002 [21] standard.

Vienna Development Method (VDM) The VDM is a formal method for computer system development, which supports modeling, testing and proving specific properties of the models and code generation from the validated models. Although VDM resembles known programming languages, systems can be modeled at a higher abstraction level than with programming languages, allowing for the analysis and design of the system, and identification of defects at early stages of development. It has a long history in industry thus a number of tools exist, both proprietary and free software, which help in the modeling and development phases.

Further reading

- Wikipedia on formal methods [22]
- Appelo, Jurgen. *Software Development Methodologies: The Definitive List* [23] Version 3. Knol. 2008 Jul 31.

- L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, second ed., Addison-Wesley, 2003
- M. Dorfman and R.H. Thayer, eds., *Software Engineering (Vol. 1 & Vol. 2)*, IEEE Computer Society Press, 2002
- K. Beck, *Extreme Programming Explained: Embrace Change*, Addison-Wesley, 1999
- S. McConnell, *Code Complete: A Practical Handbook of Software Construction*, Microsoft Press, second ed., 2004

3.3.2 Implementation environment

When the software engineering process comes to the point of actual implementation for a specific platform or environment, many practical considerations apply. These considerations vary from pure subjective preference over one environment or the other, to actual limitations, or benefits in comparison between environments. These considerations can be about interoperability, reusability, community support, cost, documentation, available tools, openness and standards compliance, previous technical expertise of the software development team and many other factors.

Only a few environments or platforms have been standardized in the software engineering industry, because the industry needs are always pressing for refined and improved versions with new or modified functionality. Such examples of standards include programming languages or environments which have met wide acceptance by the computer science community during decades, such as the C (ISO/IEC 9899:1990) and the C++ programming language (ISO/IEC 14882:2003), FORTRAN (ISO/IEC 1539-1:2004), and various versions of POSIX family of standards, which is an IEEE standard of system interfaces for the Unix operating system (e.g. POSIX:2008 or IEEE:1003.1-2008). However, some platforms are considered to be de facto standards nowadays, because they maintain a large user and developer group.

From the commercial and proprietary environments, the most prominent are the technologies of Microsoft Corporation (MS Windows [24], VisualStudio [25], .NET framework [26]) and Apple Inc. (Mac OS X [27], Xcode [28], Cocoa [29]).

From the open source, free software or community based technologies the most prominent are the various Linux (Unix-type) distributions (Ubuntu, Fedora, Debian, openSUSE, CentOS, Knoppix etc), the Java [30] programming language, the Eclipse [31] and NetBeans [32] IDEs and the LAMP [33] (Linux, Apache, MySQL, Perl/Python/PHP) collection of free, open source software.

Although the architecture and design of a system should not rely on implementation issues such as the underlying execution environment, and although the architecture definition should be made based on communication and data exchange standards and protocols, abstraction, interfaces and separation of implementation issues, many times the exact execution environment defines various architectural decisions because of the availability or not of various libraries, protocol implementations, adherence to standards, interoperability issues and other practical

reasons. Consequently, knowing the exact development and execution platform, sometimes can enhance or degrade the development process.

3.3.3 Code formatting

A widely adopted concept about source code and its formatting is that although the code will be processed by machines, it should be written and structured with the intention to be read by humans. To aim this task, from early years of software engineering there have been created some conventions about the programming style and the source code structuring, to ease the collaboration between software developers (e.g., B. W. Kernighan and P. J. Plauger, The Elements of Programming Style, McGraw-Hill, New York, 1974. ISBN 0-07-034199-0). Similar to these early guides and conventions, various modern recommendations exist, such as:

- Code Conventions for the Java Programming Language [34]
- C++ Coding Standard [35]
- Style Guide for Python Code [36]
- Perl Coding Standards [37]
- PHP Coding Standards [38]
- W3C recommendations and guidelines for HTML coding [39]
- GNU Coding Standards [40]

Another way to ensure that all code created in a project adheres to the same conventions, is to use automatic code generation tools or code beautifiers, which apply specific programming styles to source code. Most of the available Integrated Development Environments (IDEs) come with such code beautifiers, or can be extended with appropriate plugins.

3.3.4 Testing

Software testing is an activity performed for evaluating the software quality and for improving it by identifying defects and problems and consists of the verification of the behavior of a program, against the expected behavior. Consequently, we test a system or software product against its requirements and its specification, and if the specification of a system is erroneous, the testing procedure cannot identify these defects. The testing procedure can be performed in various steps, in various depths and various procedures. Testing a system's pieces in isolation is usually called unit testing. Depending on the context, unit testing could be about evaluating small units of code, large components of a program or individual subprograms of a system. When testing the interaction between software components, we refer to it as integration testing, while we define as system testing the evaluation of a system"s behavior as a whole. The tests are usually drawn from

the requirements, adapted accordingly to form test cases, and can be categorized to many categories such as:

Conformance testing Validating whether the observed behavior conforms to its specification.

Acceptance testing Checking the system behavior against the customer"s requirements. This testing may not involve the developers, but be performed directly from the customer to decide whether to accept the product.

Installation testing System testing against hardware or underlying platform requirements.

Beta testing Test of a system by real end users, before the actual release of software.

Reliability testing By randomly generated tests, based on an operational or statistical model, various measures about the reliability of a system can be derived.

Regression testing Testing a system which was previously working, whether it still works after some modifications or additions.

Performance testing Testing a system against specific performance requirements, such as speed, responsiveness, throughput, etc.

Stress testing Testing the limitations of a system, at maximum load.

Usability testing This testing evaluates how easy it is for end users to use the software, with or without proper training and documentation.

Software testing standards

- IEEE 610.12-1990 (R2002), IEEE Standard Glossary of Software Engineering Terminology.
- IEEE 829-1998, Standard for Software Test Documentation.
- IEEE 982.1-1988, IEEE Standard Dictionary of Measures to Produce Reliable Software.
- IEEE 1008-1987 (R2003), IEEE Standard for Software Unit Testing.
- IEEE 1044-1993 (R2002), IEEE Standard for the Classification of Software Anomalies.
- IEEE 1228-1994, Standard for Software Safety Plans.
- IEEE/EIA 12207.0-1996 // ISO/IEC12207:1995, Industry Implementation of Int. Std. ISO/IEC 12207:95, Standard for Information Technology-Software Life Cycle Processes.
- ISO/IEC JTC 001/SC 07/WorkingGroup 26 works in the ISO/IEC 29119 Software Testing working draft that will replace the existing standards
  - IEEE 829:2008
  - IEEE 1008:1987

3.4 Evaluation - Maintenance

The software development aims to deliver a product which satisfies user requirements. Since the user requirements change or evolve through time, defects are uncovered and technology evolves, the maintenance phase is a significant part of a product's lifecycle even though it usually does not receive as much attention as it should. The driving needs for maintenance can vary, such as correcting faults, improving the design or the implementation, interface with other systems, adaptations for different hardware and software requirements and so on. Thus, we can categorize the types of software maintenance as:

Corrective maintenance Reactive modification of a software product performed after delivery to correct discovered problems

Adaptive maintenance Modification of a software product performed after delivery to keep a software product usable in a changed or changing environment

Perfective maintenance Modification of a software product after delivery to improve performance or maintainability

Preventive maintenance Modification of a software product after delivery to detect and correct latent faults in the software product before they become effective faults

These are described in the ISO/IEC 14764-1999 *Software Engineering - Software Maintenance* standard which describes standardized techniques for software maintenance. Other standards which are related to software maintenance, evaluation or quality mtrics which affect maintenance and evaluation processes are :

- IEEE 610.12-1990 (R2002), IEEE Standard Glossary of Software Engineering Terminology.
- IEEE 1061-1998, IEEE Standard for a Software Quality Metrics Methodology.
- IEEE 1219-1998, IEEE Standard for Software Maintenance.
- IEEE/EIA 12207.0-1996//ISO/IEC12207:1995, Industry Implementation of Int. Std. ISO/IEC 12207:95, Standard for Information Technology -Software Life Cycle Processes.
- IEEE Std 14143.1-2000// ISO/IEC14143-1:1998, Information Technology - Software Measurement-Functional Size Measurement - Part 1: Definitions of Concepts.
- ISO/IEC 9126-1:2001, Software Engineering-Product Quality - Part 1: Quality Model.
- ISO/IEC 14764-1999, Software Engineering - Software Maintenance.
- ISO/IEC TR 15271:1998, Information Technology - Guide for ISO/IEC 12207, (Software Life Cycle Process).

3.4.1 Licensing

A software license governs the usage or distribution of software. With the exception of public domain material, all software are copyright protected. There are generally two types of software license: Proprietary licenses and free and open source licenses.

Proprietary software licenses are generally granted by software publishers to the user under a end-user license agreement (EULA), but the ownership is retained by the software publisher. This in effect means that the user must accept the EULA conditions if they wish to use the software.

Open source license generally can be categorized into two main groups. Copyleft licenses where the aim is to preserve the openness of the software and Permissive licenses where the aim is to give freedom to the user of that software. An example of copy left license includes GPL while permissive free licenses includes BSD and MIT license. A general characteristic of these types of free software form of licensing is that the user may use the software without accepting the license however if they wish to exercise any additional rights granted by the licenses, the user must accept and adhere to the conditions set out in the licenses.

A list of open source licenses is set out in Appendix B and they are also listed and compared on Wikipedia [41] [42].

3.5 Software Quality

The notion of quality is sometimes perceived as a matter of subjective criteria. Various definitions about the software quality exist, which can be summarized as the degree of conformance to user requirements, or the degree of customer satisfaction from the end product. Quality characteristics may be required or not, or may be required to a greater or lesser degree, and trade-offs may be made among them.

In the industry there are various standards about quality procedures and metrics which can also be, partly or fully, applied to the software engineering process, such as the ISO900 [43] family of standards about quality management, the CMMI [44] (Capability Maturity Model Integration) which deals with process improvement approaches and the ISO/IEC 15504 [45] standard (also known as SPICE - Software Improvement and Capability Determination) which is a set of technical standards for the computer software development processes and business management functions.

The quality assurance process on software development is based on verification and validation of the end product based on testing, inspections, audits, technical and management reviews, as well as adhering to standard processes which ensure quality management via the lifecycle of software. There are many practical considerations which affect the quality of the end software product, such as the particular domain of the software, its requirements, the external components which are used in the system, the methods and tools which are used during the development, maintenance and evaluation of the system, the budget, the staff, the project organization and the scheduling of all the processes. Even the defect characterization and handling can reveal the

degree of quality in a system, such as the distinction and handling between different types of defects and the so-called fault-tolerance of a system to situations such as Errors (the difference between a computed result and the correct result), Faults (incorrect step, process, or data definition in a computer program), Failures (the (incorrect) result of a fault) and Mistakes (a human action that produces an incorrect result).

The models of software product quality often include measures to determine the degree of each quality characteristic attained by the product. If they are selected properly, measures can support software quality (among other aspects of the software life cycle processes) in multiple ways. They can help in the management decision process. They can find problematic areas and bottlenecks in the software process and they can help the software engineers assess the quality of their work for longer-term process quality improvement. There are also a few topics where measurement supports software quality management directly. These include unit testing results, reliability models and benchmarks, fault and failure data, statistical tests and analysis of prediction models. Some standards which are related with software quality assessment, measurement, definition, methodology and guidelines, are:

- IEEE 730-2002, IEEE Standard for Software Quality Assurance Plans.
- IEEE 982.1-1988, IEEE Standard Dictionary of Measures to Produce Reliable Software.
- IEEE 1008-1987 (R2003), IEEE Standard for Software Unit Testing.
- IEEE 1012-1998, Software Verification and Validation.
- IEEE 1028-1997 (R2002), IEEE Standard for Software Reviews.
- IEEE 1044-1993 (R2002), IEEE Standard for the Classification of Software Anomalies.
- IEEE 1059-1993, IEEE Guide for Software Verification and Validation Plans.
- IEEE 1061-1998, IEEE Standard for a Software Quality Metrics Methodology.
- IEEE 1228-1994, Software Safety Plans.
- IEEE 1462-1998//ISO/IEC14102, Information Technology - Guideline for the Evaluation and Selection of CASE Tools.
- ISO/IEC12119:1994, Information Technology-Software Packages - Quality Requirements and Testing.
- ISO 9001:2000, Quality Management Systems - Requirements.
- ISO/IEC 9126-1:2001, Software Engineering - Product Quality, Part 1: Quality Model.
- ISO/IEC 14598:1998, Software Product Evaluation.
- ISO/IEC 15026:1998, Information Technology - System and Software Integrity Levels.

- ISO/IEC TR 15504-1998, Information Technology - Software Process Assessment (parts 1-9.
- ISO/IEC 15939:2000, Information Technology - Software Measurement Process.
- ISO/IEC 90003:2004, Software and Systems Engineering - Guidelines for the Application of ISO9001:2000 to Computer Software.

4 Human-computer interface and web standards

Relevant to WP4, WP7, WP8, WP13, WP14

Many work packages will develop human interface web servers and programmatic web services. This section describes standards which relate to this development.

4.1 Protocols

4.1.1 HTTP and HTTPS

HTTP is the basic communication protocol of the web, and its latest version (HTTP/1.1) was defined in RFC 2616 [46] in 1999. It is widely implemented in both servers and clients and is clearly the only standard which is relevant for web services, therefore details of the standard are not given here.

HTTPS, defined in RFC 2818 [47] builds upon HTTP, adding a layer of encryption to secure data against eavesdropping and man-in-the-middle attacks during transmission. It relies upon trusted certification authorities (CAs), the most popular of which are typically included with browsers. Several attacks upon the security of HTTPS have been documented, but its ubiquity and integration with current web infrastructure make it the only obvious choice for securing data destined for web browsers. It is a less obvious choice for web services interfaces, because alternatives exist according to the web services protocol being used - see section on web services.

Some p-medicine services are to be deployed on cloud infrastructure. HTTPS in its most basic form supports only one certificate per port per IP address. Therefore the presence of multiple virtual servers sharing a port on a single real server prevents the use of HTTPS. A current solution to this is Server Name Identification (SNI), but this isn't supported on very old browsers - including IE6. An alternative is to use different ports for each virtual server, or set up virtual IP addresses for each virtual server. Alternatively, for older clients, user-facing web servers might use standard non-virtualised servers.

Since most data transmitted in p-medicine will be sensitive, HTTPS should always be used, rather than HTTP.

4.1.2 Web services protocols: SOAP and REST

Although HTTP was primarily designed to transmit content for human consumption, it has been re-purposed for communication between electronic devices. A key motivation for this was to allow RPC type communication to tunnel through firewalls, which are typically configured to allow HTTP traffic.

SOAP (Simple Object Access Protocol) [48] was the first popular approach, although its transport layer was not restricted to HTTP. Messages are structured using XML with a combination of a standard SOAP envelope schema and an application-specific schema. This is contained in a GET or POST HTTP request with a SOAP-specific content type.

Many standards have built up around SOAP addressing application-specific needs. In particular, Web Services Security (WS-Security) [49] defines standards for signing and encrypting SOAP messages, using various common formats and algorithms, and implementations exist for most web services platforms.

REST (Representational State Transfer) [50] is technically and conceptually an architectural style, rather than an alternative protocol to SOAP. However, HTTP is the major architecture which conforms to the principles of REST (HTTP/1.1 and REST were developed in parallel), and when one speaks of a REST service, this typically refers to the use of HTTP and related web standards to implement RPC style communications. This is more precisely and correctly referred to as a *RESTful web service*. Technically, then, SOAP can be RESTful if it follows the design principles of REST, although they are usually spoken of as being distinct.

Less philosophically, RESTful web services make more use of all the verbs defined in HTTP. SOAP services typically use just POST (and sometimes GET) methods when using HTTP as a container, giving details of the precise nature of the request in an application-specific manner within the SOAP envelope. REST services attempt to reveal some of the semantics of a request by making use of the full HTTP verb vocabulary, URIs to refer to resources upon which verbs act, and response status codes. This allows standard web intermediaries (caches, proxies, tunnels, firewalls) to act upon messages more intelligently.

REST is the current trend in web services, although there is little evidence that the stated advantages of REST are real (although there is a great deal of opinion on the matter). Usually the justification for using REST is that there's no good reason to *not* use it. Attendees at the p-medicine workshop on standards in July 2011 seemed to agree that sticking to REST principles was a reasonable choice for our project, with the caveat that we might find security troublesome and will have to stay flexible when it comes to the details of implementation.

4.2 Text and media formats

4.2.1 Content markup languages

HTML is the dominant standard for text and content markup on the web. Whilst HTML 4.01 is the latest official standard, **HTML5** is currently in advanced stages of development and many new features in the latest draft specifications are supported by the latest web browsers. HTML5 adds many useful elements to HTML, including video, audio and canvas tags, and official inclusion of SVG (for vector graphics) and MathML (for expressing mathematical formulae) in the standard. There is an increased focus upon standardised APIs for scripting.

Lack of support in older browsers means that using HTML5 at this stage may be premature, especially given legacy systems deployed in clinical environments and the slow pace of change in those environments. As with many web standards, use of the latest standards and facilities may need to paired with a less advanced but more widely supported backup solution. At the p-medicine Standards Workshop in July 2011, Eric Prud'hommeaux recommended the use of HTML5 given the timescale of p-medicine development and deployment.

HTML is intended as a markup language for interlinked hypertext documents. As such, elements are generally presentational rather than semantic in intent. HTML has been extended to allow the inclusion of non-document semantic information, mainly using the *class* attribute of most elements. This can be turned into presentational information using CSS. **XML** can be regarded as a purer semantic representation of information. It is a generalised markup language which is not specific to hypertext documents. Whilst most web browsers can display XML documents, the lack of *any* presentational information means it is typically in a very technical browsable tree form which is not user-friendly. An XML document may be transformed into a more human-readable form with domain-specific browsers, or more generally using an XSLT description (see next section).

XHTML is a parallel standard to HTML, designed to mirror all of its functionality but using XML as its basis rather than the more flexible SGML. XML, and therefore XHTML, is somewhat stricter and more consistent on how tags, attributes and content are arranged, and ambiguous formatting is reduced. All XML documents are valid SGML documents, but not vice-versa. There are XHTML versions to mirror HTML 4.01 and HTML5. All browsers which support normal HTML also support the equivalent XHTML, since it is a subset of the standard. XHTML documents can be parsed with an XML parser, which is relatively simple compared to a general HTML parser which must deal with ambiguous markup.

4.2.2 Content styling and presentation languages

CSS (Cascading Style Sheets) describes the visual presentation of HTML documents. CSS descriptions can be included in the *style* attribute of an HTML tag, but this is generally regarded as bad practice. It is better to separate style from the content of an HTML document. The style for a set of HTML tags which match a certain pattern can be placed in the header of an HTML

document, or in a separate CSS file which the document links to. A set of such patterns is termed a stylesheet, and several stylesheets can be associated with a given HTML document, which can be chosen according to user taste, accessibility requirements, or the presentation medium (for example, a standard computer screen, a small-format smartphone, a projector, or a hard-copy printer). CSS is mostly supported in all modern browsers, with a few omissions in the less-used functionalty. For example, support for paged media (hard-copy) is often incomplete.

XSLT (Extensible Stylesheet Language Transformation) is a declarative XML-based language for transforming an XML document into some other form, most often HTML, but possibly any other textual format. Although it is a more general tool than CSS, its primary intent is to transform XML documents into a more user-friendly human-readable form. Most web browsers support XML transformation via XSLT on the client side. Additionally, an XSLT transformation may be applied server-side so that only HTML is served. XML in combination with XSLT therefore represents the most flexible and pure solution to marking up content semantically, but presenting it in a user-friendly manner. The key disadvantages are is that developers may be unfamiliar with this paradigm, XSLT is a fairly verbose and ugly language which is difficult to develop and debug, and it is not a commonly used paradigm and therefore there maybe issues regarding its robustness and future support.

4.2.3 Static imagery

Two recurring and related issues affect many non-textual media formats. Such formats, mostly those involving compression, sometimes involve patented algorithms and may have only proprietary codecs available. As a result, codecs are not available, either for fear of violating patents, or because a proprietary codec is not available for a particular platform.

The established standards for storing and transmitting raster imagery for web browsers are **PNG** for lossless encoding, and **JPEG** for lossy encoding of photographs. PNG is a flexible format which was designed using unpatented algorithms, to replace the widely used GIF format. PNG is universally supported by web browsers. JPEG includes patented algorithms and legal proceedings regarding these are ongoing. No suitable alternative lossy format exists, and JPEG is so widely used and supported it seems unlikely that any patent action will prevent its use in the near future.

Some imagery in p-medicine might be very high resolution, and require an interface for zooming in to details. Several solutions exist for this. For images up to 2-3 times the size of a display, many browsers deal with this be scaling the image to the screen size, and allowing a click to expand it to full size. This becomes cumbersome and wasteful for very large images. The **HTML5 *canvas* tag**, which allows setting of pixel coluor values within a rectangular area, maybe become a useful method for presenting very large images in the future. The **Google Maps Javascript API**, which stores pre-rendered tiles of a very large images at different scales, and transmits them on demand, is available for third parties to present any large-scale imagery as custom map tiles.

Finally, some images may be expressible in a vector format. For example, annotation markings on medical imagery, or image segmentation data. Currently there is no clear and widely

well-supported method for displaying such imagery and allowing zooming and panning. **SVG** was established some time ago as the vector graphics equivalent of HTML, but support in browsers is often poor. This is set to change since it was adopted as part of the HTML5 standard. Vector images may also be presented in Flash or Java plugins, which are discussed later.

4.2.4 Time-based media: video and audio

The *de facto* standard for video on the Web, thanks to YouTube, is **Flash**, which presents video encoded in a Flash container format (**FLV**) in a Flash plugin with custom controls. Whilst this is very widely supported, Apple have omitted support for it from all of their non-desktop products (including the iPod touch, iPhone and iPad). Special provision must be made for these platforms, with video served in **H.264** format. H.264 and most of the encodings available within FLV are patented.

HTML5 aims to choose at least one standard video encoding for the web, so that platform-specific issues can be ignored. The actual format is the subject of current debate between vested interests in industry. An initial candidate was **Ogg Theora**, and this is currently the most widely supported format in browsers out-of-the-box (notably excepting Safari). But it is not part of the HTML5 draft standard yet.

The situation with audio is similar. Although **MP3**, **Ogg Vorbis** (lossy) and **WAV** (non-lossy) are widely supported standards for audio generally, the interface presented for playing audio files in web browsers is generally poor or non-existent, usually relying upon plugins or custom Javascript controls. Again, a common approach is to use Flash to present an interface to an audio player. Some of these are generally available and free to use, for example SoundCloud [51]. HTML5 is also attempting to standardise at least one audio encoding, initially Ogg Vorbis, but the issues with video standardisation are replicated

It seems likely that multiple formats will be necessary to support audio and video on a wide range of devices, if these are deemed a necessary part of p-medicine.

4.2.5 Domain-specific formats

Many domain-specific files might have to be presented to the user. There are many approaches to doing this.

Most commonly, a file will be processed and converted on the server side into a representation widely supported by browsers - HTML, CSS, images, video and audio. This is the preferred method where such representations are adequate.

Often a more advanced domain-specific client-side browser will be desirable, particularly where 3D or advanced layout and interaction is needed. Often the approach is then to use Flash or Java plugins, or even a purpose-built plugin. The disadvantage here is that plugins might not be available for particular platforms, or installed on a particular device. Most such cases might be

avoided by use of the latest facilities of HTML5, particularly the *canvas* tag. Such decisions will have to be made on a case-by-case basis, according to the complexity of the presentation and the availability of existing solutions.

4.3 Client-side execution

Many of the solutions above rely upon execution of downloaded code on the web client. Client-side execution has several key advantages:

- Allows a quicker response to user action than is possible when relying purely upon client-server communication.
- Some solutions allow more flexible layout and drawing, direct access to a region where arbitrary objects may be rendered
- Some solutions allow more flexible interaction capabilities, with direct access to mouse and keyboard events

These are all potentially useful for making p-medicine's web interfaces more usable and powerful. The second two advantages here are addressed with the HTML5 *canvas* tag. A key disadvantage of client-side execution is that it can create profound accessibility and usability problems if these are not explicitly addressed. This is mainly because the built-in accessibility facilities of HTML are not exploited or are broken. Additionally, sometimes the plugins required to execute code on the client are not available on a device.

Javascript is an interpreted scripting language typically used to create more slick and responsive interfaces from standard HTML components. It can access and alter many elements of an HTML page, most frequently form elements, through a DOM API. It can also perform requests background to the web server without having to request a new web page, giving pages a more interactive feel. It was initially developed as a proprietary scripting language, but was quickly adopted with variations by many browser makes, and it became a *de facto* standard. The variations in language and API meant Javascript code often has many variants based upon detection of browser and browser version. It is now to be included in a standardised form in HTML5, and browsers are racing to implement more efficient interpreters, motivated in part by Google's heavy use of it. All of these are signs that Javascript the best choice for client-side execution, if it cannot be avoided, or if there is a truly compelling usability case in favour of it. The main reason that Javascript might not be available on a device is if it has been explicitly disabled by the user or by systems administrators, for security, privacy or reliability reasons.

While Javascript standardisation is being achieved, **Google's Javascript API** offers a library of code which hides the platform-specific elements and allows rapid development of fairly rich applications, but still within the constraints of the current HTML standards.

Three different platforms compete for the full *rich internet applications* market, which allow desktop-style interaction with advanced and flexible layout and rendering facilities. In terms of

technology and standardisation, they are all proprietary, have similar functionality, are quite mature and preference is a matter of developer taste. The key difference is market penetration, which is assessed here at statowl.com [52].

Flash is intended primarily as a medium for adding flexible graphics, audio and interaction capabilities to web pages, although it includes a complete language which allows arbitrarily complex applications to be implemented. Its foundation in time-based media, however, makes Flash development unfamiliar to developers from a C++ or Java background. The Flash development environment *Adobe Flash Builder* is not free, however Flash applications can be developed using in a limited fashion using the open source Flex SDK, which can be integrated with a third party development environment such as Eclipse. It is available on most platforms, notably excepting Apple's mobile devices. The Statowl survey finds around 95% of deployed web clients support Flash.

Java was an early attempt at executing arbitrary code on the client in the form of applets. Today, its use for this is uncommon, and Java has moved server-side, onto mobile devices, and embedded systems. This means that the Java plugin is often not installed in the browser. Only Google Chrome and Safari on Mac OS X have built-in support for Java. The Statowl survey finds around 77% of deployed web clients support Java.

Silverlight is Microsoft's answer to Flash. It is made available as a plugin by Microsoft on Windows and Mac OS, and by Novell as Moonlight on Linux, although usually supporting an older version of the Silverlight specification than the official Silverlight implementation. The Statowl survey finds around 64% of deployed web clients support Silverlight.

4.4 Usability and accessibility

Usability and accessibility are closely related concepts. **ISO 9241** defines many standards relating to the *ergonomics of human-computer interaction*, and defines three key elements of usability:

- Effectiveness
- Efficiency
- Satisfaction

Ergonomics and usability standards usually refer to average users within certain "normal" boundaries and limits. Accessibility refers to usability of systems by users outside of these norms, for example those with physical impairments, or those operating in unusual environments. The two concepts are interrelated, and often systems developed to enhance accessibility also enhance general usability. Accessibility in p-medicine is particularly important, since certain web interfaces for patient empowerment are likely to be used by people suffering some physical impairment, or clinicians working on non-desktop devices or with time and concentration constraints. The ISO 9241 set of standards appear to be very relevant to our project, but they cost money to access.

W3C have defined the **Web Content Accessiblity Guidelines** [53], currently at version 2, as part of the broader **Web Accessibility Initiative** [54]. The guidelines include advice on how to interpret them and how to follow them using current web technologies, and are structured as follows:

- Perceivable
  - Provide text alternatives for any non-text content so that it can be changed into other forms people need, such as large print, braille, speech, symbols or simpler language
  - Provide alternatives for time-based media
  - Create content that can be presented in different ways (for example simpler layout) without losing information or structure
  - Make it easier for users to see and hear content including separating foreground from background
- Operable
  - Make all functionality available from a keyboard
  - Provide users enough time to read and use content
  - Do not design content in a way that is known to cause seizures
  - Provide ways to help users navigate, find content, and determine where they are
- Understandable
  - Make text content readable and understandable
  - Make Web pages appear and operate in predictable ways
  - Help users avoid and correct mistakes
- Robust
  - Maximize compatibility with current and future user agents, including assistive technologies

It is clear that most of the ideas actually improve core usability, and not just accessibility. Concern was expressed at the p-medicine Standards Workshop in July 2011 that these guidelines were practically impossible to follow in their entirity. Nonetheless, they seem like a sensible set of guidelines to aspire to, if not adhere to.

The USA Government maintains a set of **guidelines for developers of USA Government websites** [55] which give concise descriptions of guidelines, along with importance ratings and ratings of evidence that show a guideline is well-founded, with references to research. This latter element makes these guidelines particularly authoritative, although they are perhaps a little out of date, and perhaps US-centric in that they do not deal with internationalisation issues, and appeal more to a US design aesthetic.

The **Web Style Guide** [56] is a respected authority on web design, and is as much about aesthetics as usability and accessibility. Although aesthetics must never impinge upon usability or accessibility, good aesthetics can certainly enhance both, particularly in fulfilling the *Satsfaction* criterion of the ISO standard. Such guidelines are useful to us on p-medicine since no formally trained designers are involved in the project. The best approach, perhaps, is to ensure we make web output as plain as possible, with a standardised semantic markup, and allow the possibility that a designer can be employed to apply a good and consistent visual style on all devices and across all p-medicine web interfaces.

5 Data structure and semantic standards

This section is a part of deliverables for WP4.

6 Data exchange formats and protocols

Relevant to WP4, WP7, WP11, WP12

6.1 Clinical data

With the increased adoption of electronic patient records there is an increase opportunity for creating longitudinal patient records that span many decades and aggregate data from multiple institutions and for collaboration among healthcare organizations in the process of delivering care. Within this trend the need for standards for the representation and exchange of patient data has become apparent.

As defined by the Health Information Management Systems Society (HIMSS):

> "The Electronic Health Record (EHR) is a longitudinal electronic record of patient health information generated by one or more encounters in any care delivery setting. Included in this information are patient demographics, progress notes, problems, medications, vital signs, past medical history, immunizations, laboratory data, and radiology reports. The EHR automates and streamlines the clinician's workflow. The EHR has the ability to generate a complete record of a clinical patient encounter, as well as supporting other care-related activities directly or indirectly via interfaces including evidence-based decision support, quality management, and outcomes reporting."

EHRs use both technical and clinical standards. However, EHR vendors have only implemented some standards, while having a great deal of variation in their implementations, which results in systems that cannot interoperate and for which secondary use of data for research, epidemiology,

etc. is difficult. Current EHR systems, due to their evolution over time, are often just an electronic representation of the previously used paper records.

> "Electronic patient records today are highly idiosyncratic, vendor-specific realizations of patient record subsets. They adopt few, if any, health information standards, and very rarely accommodate controlled terminologies where they might be sensible. The reason for this epidemic of incompatible data has more to do with the limitations of available information standards and machineable vocabularies than with any fundamental unwillingness to adopt standards. A compelling business case, for system vendors or patient providers, simply has not emerged to foster standards adoption and systems integration."

According to the report of the National Institutes of Health (National Center for Research Resources) report on **EHRs** [57], three main organizations create standards related to EHRs: Health Level Seven (HL7), Comite Europeen de Normalization - Technical Committee (CEN TC) 215, and the American Society for Testing and Materials (ASTM) E31. HL7, which operates in the United States, develops the most widely used healthcare-related electronic data exchange standards in North America. CEN TC 215, which operates in 19 European member states, is the preeminent healthcare IT standards developing organization in Europe. Both HL7 and CEN collaborate with the ASTM, which operates in the United States and is mainly used by commercial laboratory vendors.

6.1.1 HL7 Reference Information Model

The main aim of the HL7 messaging standard is to ensure that health information systems can communicate their information in a form which will be understood in exactly the same way by both sender and receiver. Whereas HL7 version 2 was a pure messaging standard for interoperability, version 3 (V3) not only specifies how to send a message, but also what a message can contain. To achieve this goal, V3 makes use of vocabularies and ontologies like SNOMED and LOINC. At the basis of all HL7 V3 messages is the Reference Information Model (RIM), an abstract model of the concepts which underlie healthcare information.

The RIM is defined as an object-oriented model, and the following are the definitions of the six core HL7 RIM classes:

Act an action of interest that has happened, can happen, is happening, is intended to happen, or is requested/demanded to happen. An Act instance is a record of such an action.

Entity a class or specific instance of a physical thing or an organization/group of physical things capable of participating in Acts. This includes living subjects, organizations, material, and places. The Entity hierarchy encompasses human beings, organizations, living organisms, devices, pharmaceutical substances, etc.

Role establishes the roles that entities play as they participate in an Act. Each role is "played" by one Entity (the Entity that is in the role).

Participation an association between a Role and an Act. The Participation represents the involvement of the Entity playing the Role with regard to the associated Act. A single Role may participate in multiple Acts and a single Act may have multiple participating Roles.

ActRelationship an association between a pair of Acts. This includes Act-to-Act associations such as collector/component, predecessor/successor, and cause/outcome. The class has two associations to the Act class, one named *source* the other named *target*.

RoleLink a connection between two roles expressing a dependency between those roles

6.1.2 HL7 Clinical Document Architecture

The HL7 Clinical Document Architecture (CDA) is a document markup standard, based on the HL7 RIM, that specifies the structure and semantics of clinical documents for the purpose of exchange. Examples of such clinical documents are referral notes, discharge summaries, and clinical summaries. The CDA document is defined such that it can include text, images, sounds, and other multimedia content. A CDA document is encoded in the XML language, and basically consists of a header and a (structured) body. Examples of such clinical documents are referral notes, discharge summaries, and clinical summaries. The CDA document is defined such that it can include text, images, sounds, and other multimedia content. A CDA document is encoded in the XML language, and basically consists of a header and a (structured) body. The header classifies the document and provides information on the encounter, the patient, and other involved entities. The body contains the actual clinical report, and often is divided into nestable document sections. Each of these sections can contain a single "narrative block" (that is, human readable content) and any number of CDA entries. These entries represent structured content and can be an envelope for information described in each of the HL7 domains, as the CDA is RIM-compliant.

6.1.3 IHE

IHE Integration Profiles [58] describe the solution to a specific integration problem, and document the system roles (Actors), standards and design details for implementers to develop systems that cooperate to address that problem. **IHE** has introduced the first standards-based approach to cross-enterprise exchange of health information through its Cross-Enterprise Document Sharing (XDS) integration profile. XDS provides a specification for managing the sharing of documents between any healthcare enterprises, and handles a broad range of clinical documents including diagnostic imaging, medical summaries and scanned documents leveraging established standards such as DICOM and HL7. With the iHistory platform, HxTI provides a certified, XDS-compliant infrastructure for interoperable health information exchange. iHistory includes the ability to transform legacy PACS and RIS into XDS-compliant sources of clinical data.

6.1.4 openEHR

openEHR is an open, detailed, and tested specification for a comprehensive interoperable health information computing platform for the EHR. It is based on a two level methodology concerning the development of information systems that separates the semantics of information and knowledge into two levels. The Reference Model corresponds to the information level and consists of a relatively small number of non-volatile abstract concepts. At the knowledge level models defining domain concepts by expressing constraints on instances of the underlying Reference Model are built, called archetypes.

Next to the use of standards for data representation and exchange, the use of standard clinical vocabularies and of widely adopted ontologies would greatly enhance the ability of clinical information systems, such as EHRs to interoperate in a meaningful way. While syntactic interoperability is essential, the real added value with respect to automatically understanding the meaning of data from different systems, reasoning about the data and integrating that data into meaningful applications will come from enabling semantic interoperability.

6.1.5 CDISC

The most relevant data-related standard in clinical research is **CDISC** [59] Currently, CDISC is the leading standards development organization for the medical research domain. Its mission is to develop industry standards to support acquisition, exchange, submission and archiving of clinical trials data for medical and biopharmaceutical product development. The various standards are based on the CDISC Operational Data Model (ODM) standards. The standards wildly vary in maturity and uptake into practice. According a communication (Feb 2010) from CDISC, the FDA's Center for Drug Evaluation and Research (CDER) and Center for Biologics Evaluation and Research (CBER) give the following advice:

1. Use CDISC Standards (SDTM and ADaM) NOW in eSubmissions to CDER.
2. If you want high quality eSubmissions (which will be reviewed better and more quickly), start with CDISC CDASH for your case report forms when you plan your study.
3. The expected transport format for the foreseeable future is SASXPT with Define.xml

In order to facilitate an efficient submission of trial results to regulatory bodies, CDISC has defined the Study Data Tabulation Model (SDTM). The SDTM is a general framework describing the organization of the information that is collected during clinical trials. The SDTM consists of a set of clinical data file specifications and underlying guidelines. These different file structures are referred to as domains. Each domain describes a type of data associated with clinical trials, such as demographics, vital signs or adverse events. CDISC also provides a standard for running a clinical trial, namely the Operational Data Modelling (ODM) standard. ODM supports interchange between applications used in collecting, managing, analysing and archiving data.

ODM provides a format for representing study metadata, study data and administrative data associated with a clinical trial. Various clinical trial management systems like SAS , OracleClinical support the ODM standard for data exchange.

ODM provides a format for representing study metadata, study data and administrative data associated with a clinical trial. It represents the data that would be transferred between different software systems during a trial, or archived after a trial. There are two types of ODM files: snapshots files and transactional files. A snapshot file is completely self-contained, containing the current state of the source database. A transactional file shows, for each included entity, both the latest state of the source database, and (optionally) some prior states of that entity in the source database. The transactional files can provide an audit trail.

Clinical Data Acquisition Standards Harmonization (CDASH) is focused on data collection (as opposed to data reporting). The CDASH project identifies the basic data collection fields needed from a clinical, scientific and regulatory perspective to enable more efficient data collection at the investigative sites. The CDASH data collection fields (or variables) can be mapped to the SDTM structure. When the data are identical between the two standards, the SDTMIG variable names are presented in the CDASH domain tables. In cases where the data are not identical, CDASH has suggested new variable names. The CDASH recommendation also includes some data collection fields that are not included in the SDTMIG, these collection fields are intended to assist in the cleaning of data and in confirming that no data are missing. In some instances the optimal data collection method conflicts with the SDTM for reporting data, in these cases additional transformations and derivations may be needed to create the final SDTM compliant datasets.

Some of the **caBIG** results, such as Common Data Elements (used for exchanging annotation, for example) need to be followed. There are also tools for CDEs. These are used in the clinical research community linked to caBIG and have the potential to become de-facto standards. For clinical research **SAS** [60] may also be relevant due to its adoption by the pharmaceutical industry.

6.1.6 Minimum Information About a Microarray Experiment (MIAME)

Although increasingly used for gene expression data analysis at a genome-wide level, microarray technology still has the limitation of insufficient standardization for presentation and exchange of such data.

MIAME [61] aims at establishing a common way for recording and reporting microarray-based gene expression data, and proposes the minimum information required to ensure that microarray data can be interpreted and that the results that yield from the analysis of the data can be independently verified. The standard only defines the content and the structure of the information and does not address the actual technical format of storing and communicating the data.

MIAME has also identified the need for controlled vocabularies and ontologies for data representation in order to enable interoperability. As there is a very limited availability of controlled vocabularies, MIAME proposes a representation in lists of *qualifier, value, source* triplets,

which authors can use to define their own attributes (i.e. qualifiers) and provide the appropriate values and the source from which the terms were extracted.

A significant amount of context data is necessary to describe a microarray experiment be-cause the results of such an experiment (gene expression) are only meaningful in the context of the conditions in which the experiment was run. Most microarray experiments only report relative changes in gene expression relative to a non-standardized reference, the data is normalized in different ways and is represented in non-standardized formats, and the annotation describing the data is often insufficient. All these factors make comparing data from distinct experiments very difficult. MIAME attempts to alleviate these issues by specifying the annotation necessary to properly interpret the data and the detailed description of the experiment, including the way in which the gene expression level measurements were obtained.

Next to the gene expression matrix, which contains for each gene and sample in the array the measured expression, MIAME advises to provide information about the genes whose expression was measured, and about the experimental conditions under which the samples were taken. The information required can be divided on a conceptual level into three logical parts: gene annotation, sample annotation and a gene expression matrix.

According to MIAME, there are at least three levels of data relevant to a microarray experiment:

1. The raw data (scanned images)
2. The quantitative outputs from the image analysis procedure (microarray quantita-tion matrices)
3. The derived measurements (gene expression data).

As the transformation steps leading from the raw data to the gene expression data are not standardized it is necessary to record a detailed description of how the expression values were obtained. The gene annotation part will provide full and detailed description of each element on the array and the sample annotation will describe the biological samples and the exact conditions in which the samples were taken. In the case of commercial arrays the required information needs to be provided only once, by the array supplier and subsequently referenced by the users. This also holds for standard protocols, which need to be described only once after they are established.

According to MIAME a microarray experiment is defined as a set of one or more hybridizations, each of which relates one or more samples to one or more arrays. The minimum information required for a published microarray-based gene expression experiment includes descriptions of the following six sections:

Experimental design the set of hybridization experiments as a whole that addresses a common biological question. This section includes a free text format description of the experiment or a link to an electronically available publication. The minimal information included in this section includes:

1. The type of the experiment
2. The experimental variables, including parameters or conditions tested
3. Quality-related indicators such as usage and types of replicates
4. The experimental relationships between the array and the sample entities. Each of these will be assigned unique identifiers that are cross-referenced with the information in the following sections. This information will allow the user to reconstruct the experimental design and to relate information from the other MIAME sections.

Array design each array used and each element (spot, feature) on the array. This section provides a systematic definition of all arrays in the experiment, including the genes and their physical layout on the array. This section contains two parts:

1. A list of the physical arrays providing for each a unique id and a reference to a particular array design
2. The description of the array designs: a description of the array as a whole (e.g. platform type, provider and surface type), a description of each type of element or spot used (e.g. synthesized oligo-nucleotides) and a description of the specific properties of each element (e.g. the DNA sequence).

Samples samples used, extract preparation and labelling. This section describes the biological material for which the gene expression profile is being generated, and is divided into three parts which describe the source of the original sample, the technical extraction of the nucleic acids and their subsequent labelling.

Hybridizations procedures and parameters. This section describes the laboratory con-ditions under which the hybridizations were carried out.

Measurements images, quantification and specifications. This section describes the actual experimental results. It consists of three parts: the original scans of the array, the microarray quantification matrices based on image analysis and the final gene expression matrix after normalization and consolidation from possible replicates. The image data should be provided as raw scanner image files and should be accompanied by in-formation that includes relevant scan parameters and lab protocols. The quantification matrices should be accompanied by a description of the software used, the underlying methodology (such as algorithms and statistics), all relevant parameters and the definitions of the quantifications used (e.g. mean or median intensity). The gene expression matrix (summarized information) consists of sets of gene expression levels for each sample. At this point the expression values may have been normalized, consolidated and transformed in any number of ways. Detailed specifications should be provided of all numerical calculations that were applied to the unprocessed quantifications to obtain the expression data.

Normalization controls types, values and specifications. This section describes the normalization strategy (e.g. spiking, housekeeping genes, etc.), the normalization and quality control algorithms, the identities and locations of the array elements serving as controls, their type (e.g. spiking, normalization, negative or positive hybridization controls,

etc.) and hybridization extract preparation (how the control samples are included in sample targets prior to hybridization). MIAME requires the storage of vast amounts of information, but often the majority of information is similar for many experiments. The goal of this standard is to describe the data in sufficient detail to allow for the understanding of how conclusions were reached. This standard only refers to the conceptual content of the data and other standards, such as MAGE-OM, MAGE-ML and MAGE-TAB, needed to be defined to encode the data for standardized acquisition, storage and interoperable communication.

6.1.7 MAGE

While MIAME focuses on the conceptual content of the data, specifying what information is needed in order to be able to interpret and reproduce a microarray experiment, MAGE provides data exchange standards to facilitate the exchange of gene expression data. The core of MAGE is the MAGE-OM, which provides an object model for the exchange of gene expression data. MAGE also provides two data exchange formats, MAGE-ML, which provides a mark-up language, and MAGE-TAB, which provides a tabular format (which is the current recommendation).
MAGE-OM defines the object model for Gene Expression data and it is modelled using UML. The model can express microarray designs, microarray manufacturing information, microarray experiment setup and execution information, gene expression data, and data analysis results, satisfying the MIAME requirements. MAGE-OM tries to be generic and as complete as possible. Users typically use a subset of the provided classes and relations, which would fulfil their needs. MAGE-ML captures MAGE-OM in an xml notation, and explicit mapping rules map the MAGE-OM model to xml. Although MAGE-ML is supported in various tools as import and export format, it is a cumbersome format to use in a laboratory when no appropriate tooling or software development expertise is available. MAGE-TAB fills this gap by providing a simple format, still capturing the requirements of the MIAME standard. MAGE-TAB is a tabular format that can be easily manipulated with various tools (even spreadsheet programs). MAGE-TAB specifies 4 type formats: Investigation Description Format (general information about the investigation), Array Design Format (array design description for each array type), Sample and Data Relation Format (relationships between samples, arrays, data and other objects), and raw and processed data files (data-matrix or the native formats of the data).

6.1.8 HL7 Clinical Genomics

In the Clinical Genomics working group, HL7 V3 standards are developed that enable the exchange of interrelated clinical and personalized genomic data. Currently, the domain consists of three main topics: Genotype, Genetic Variation, and Pedigree (Family History). The latter topic aims at describing a patient's pedigree based on genomic data. As such, it utilizes the models from the Genotype topic to contain the genomic data for the patient's relatives. The Genotype topic consists of two (HL7 RIM-based) models: the Genetic Locus and the Genetic Loci. The latter model groups several genetic locus instances, e.g. in case of a genetic test of several genes. The first model describes data related to a genetic locus: the position of a particular given sequence in

a genome or linkage map. The model includes sequencing and expression data, and can be linked to clinical information or phenotypes. Existing bioinformatics mark-up languages such as MAGE-ML and BSML are utilized to represent the raw genomic data.

The Genetic Variation topic defines a model that is a constraining of the Genotype topic models. The focus of this model is on variations in the DNA of individuals, derived using methods such as SNP probes, sequencing and genotype arrays that focus on small scale genetic changes. However, gene expression analysis, e.g. based on microarray data, is not suitable for the Genetic Variation model and will be addressed by different models within the HL7 Clinical Genomics working group.

As the costs of generating genomic data, such as microarray-based gene expression, and extracting information from that data is still quite high it does not make economic sense to discard all the collected data (and to preserve only the test result) instead of storing and re-using it, especially that it is assumed that there is much more information in the data than what can be obtained in a single test or experiment. New standards for storing and exchanging genomic data such as MIAME and MAGE also require that all data should be preserved and annotated, including the raw microarray image. Of course, MIAME's and MAGE's main focus is research, but the reasoning behind the storage and full annotation of all genomic data is precisely based on the fact that the data encapsulates information far beyond the scope of a single experiment and should be preserved and shared. In this context, there is no reason to miss on such a valuable source of data that is represented by clinical practice. Many research-focused healthcare organizations have already identified this opportunity, and they invest in infrastructure to support this approach.

6.2 Output from modelling and data mining

Significant efforts to standardise the representation of computational models in several areas of biology, such as the Systems Biology Markup Language SBML (Hucka et al., 2003) and CellML (Lloyd et al., 2004) aim to increase the exchange and re-use of models. However, the description of the structure of models is currently not suffcient to enable the reproduction of simulation results. The procedures to which the models are subjected also need to be described, as defined by the Minimum Information About a Simulation Experiment (MIASE) - see Section 6.2.1 for more details.

Large inititatives such as Sage Bionetworks [62] have also emerged having as a mission to bring together researchers and their data and knowledge, build tools and infrastructures enabling sharing and collaboration, support reuse of data, models and tools, and promote common standards and interoperability.

6.2.1 Minimum Information About a Simulation Experiment (MIASE)

MIASE [63] is a community effort aiming to identify the minimal information about a simulation experiment that is necessary in order to enable the reproducibility of simulation experiments. The proposed standard only defines the content and the structure of the information that should be provided in the description of a simulation experiment and does not address the actual technical format of storing and communicating the data.

The MIASE recommendations list the common set of information a modeller needs to provide in order to enable the execution and reproduction of a numerical simulation experiment, derived from a given set of quantitative models. MIASE aims to enable different groups to collaborate and exchange simulation experiments on computational models in biology. The goal of the project is to produce a set of guidelines suitable for use with any structured format for simulation experiments. As such, MIASE is designed to help modelers and software tools to exchange their simulation settings.

MIASE is a registered project of the MIBBI (Minimum Information for Biological and Biomedical Investigations). The guidelines are composed of three parts: Information about the models to use, Information about the simulation steps, and Information about the output. MIASE is supported by the Simulation Experiment Description Markup Language (SED-ML), an XML-based format for encoding simulation experiments.

6.2.2 Simulation Experiment Description Markup Language (SED-ML)

The Simulation Experiment Description Markup Language (SED-ML) [64] is an XML-based format for the description of simulation experiments. It serves to store information about the simulation experiment performed on one or more models with a given set of outputs. SED-ML compliant simulation descriptions will enable the exchange of simulation experiments across tools. Information on SED-ML can be found on http://sed-ml.org. The SED-ML XML Schema, the UML schema and related implementations, libraries, validators and so on can be found on the SED-ML sourceforge project page [65].

6.2.3 Data mining standards

There have been significant efforts to define standards for data mining. These are described in detail in Section 10.4.

6.3 Other data

6.3.1 Imaging data: Digital Imaging and Communications in Medicine (DICOM)

The DICOM Standards Committee create and maintain international standards for communication of biomedical diagnostic and therapeutic information in disciplines that use digital images and associated data. The goals of DICOM are to achieve compatibility and to improve workflow efficiency between imaging systems and other information systems in healthcare environments worldwide. DICOM is a cooperative standard. Connectivity works because vendors cooperate in testing via either scheduled public demonstrations, over the Internet, or during private test sessions. Every major diagnostic medical imaging vendor in the world has incorporated the Standard into its product design, and most are actively participating in the enhancement of the Standard. Most of the professional societies throughout the world have supported and are participating in the enhancement of the Standard as well.

DICOM is used or will soon be used by virtually every medical profession that utilizes images within the healthcare industry. These include cardiology, dentistry, endoscopy, mammography, ophthalmology, orthopedics, pathology, pediatrics, radiation therapy, radiology, surgery, etc. DICOM is even used in veterinary medical imaging applications. DICOM also addresses the integration of information produced by these various specialty applications in the patient's Electronic Health Record (EHR). It defines the network and media interchange services allowing storage and access to these DICOM objects for EHR systems.

6.4 Web-enabled data

The World Wide Web Consortium (W3C) produces protocols and standards for Web technologies. The World Wide Web Consortium (W3C) promoted widely adopted standards for representation of web-enabled data and knowledge: Resource Description Framework (RDF) is the de-facto model with two commonly used serializations, and is by now finding wide deployment. RDF and related semantic web services are described in detail in Section 9.5.

7 Legal and ethical standards

Relevant legal and ethical standards or guidelines will be described in WP5.

8 Security standards

Relevant to WP4, WP7, WP8, WP10, WP11, WP12, WP13, WP14

High security and privacy are necessary in modern medical applications. Strict policies are

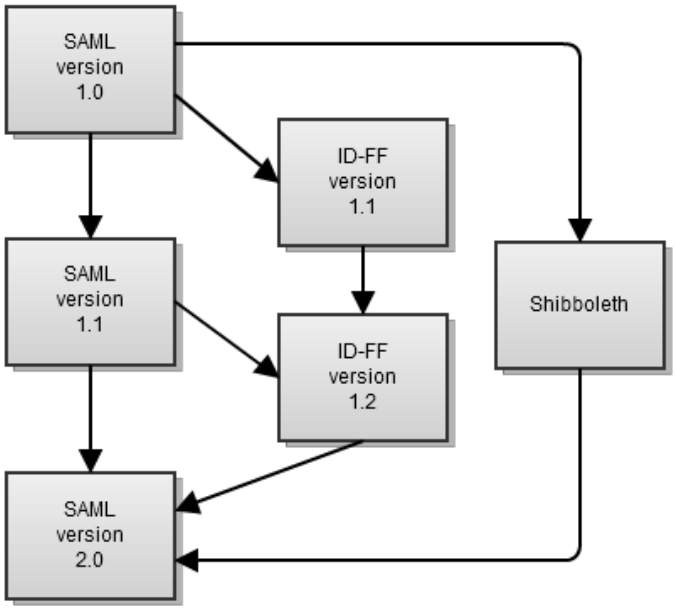

Figure 1: SAML 2.0 is a combination of three predecessor standards

defined by official medical instances to ensure that the confidential medical data can be accessed and manipulated in a secure way. For this the data is protected by different security and privacy mechanisms like authentication, identification, authorisation, anonymisation, protected data transport and storage. It is clear that the p-medicine platform must use these mechanisms to reach a high level of security.

8.1 Identity Management

8.1.1 SAML

The Security Assertion Markup Language (SAML) [66] defines an XML-based protocol, making it possible to exchange authorisation and authentication data between one or more security domains (wikipedia [67]). This exchange is done by using signed assertions containing identity information. The entity that provides the assertions is called the asserting party while the relying party is the entity that consumes and verifies the assertions. A level of trust is required between the assertion providers and the relying parties. The current version of SAML is 2.0, which is a combination of three predecessor identity federation standards: SAML 1.1, ID-FF 1.2 and Shibboleth (Figure 1). This resulted in SAML 2.0 not being compatible with SAML 1.1.

SAML mainly focusses on solving the problem of web browser single sign-on. For this it offers a single sign-on profile. In the case of single sign-on the assertion provider takes the role of Identity

Provider (IdP) and the relying party the role of Service Provider (SP). When the user visits a protected Service Provider, the SP sends the user to the IdP with an authentication request (SP-initiated web single sign-on (SSO) opposed to IdP-initiated web SSO). Once the user has successfully authenticated himself (assuming he/she is already registered to the IdP) a signed assertion is generated that contains all the information that is requested by the SP to authenticate and authorise the user. This assertion is send back to the SP which will then use it to determine whether the user is allowed to access the service. SAML does not define how a user should authenticate himself on the IdP. It leaves the responsibility completely to the security domains.

Next to the single sign-on profile, SAML offers some other useful profiles [68]:

- Single logout profile: this profile specifies how the user single sign-on session can be destroyed on the IdP and all SP's in which the user has a session running
- Attributes profile: this profile specifies how attributes with information about the user can be exchanged between the IdP and a SP.
- Name identifier mapping profile: defines a Name Identifier Mapping protocol for mapping a user's name identifier into a different name identifier for the same user.
- Assertion query/request profile: defines a protocol for requesting existing assertions by reference or by querying on the basis of a subject and additional statement-specific criteria.

SAML is widely adopted thanks to its focus on federation enablement and exchange of asserted information.

8.1.2 Liberty Alliance

The Liberty Alliance [69] is an effort of more than 150 organisations that try to establish open standards, guidelines and best practices for identity management. The main keywords in the project are federated identity, single sign-on, global logout, circle of trust and Web Services. The Liberty Alliance Project contains three main components, explained in the following three paragraphs.

The **Liberty Identity Federation Framework** (ID-FF) specifies core protocols (single sign-on (SSO), single logout (SLO), federation, name registration), schemata, bindings (HTTP and SOAP) and concrete profiles (Browser/Artifact, Browser/Post,...) that allow implementers to create a standardised, multi-vendor, identity federation network. It enables identity federation and management through features such as identity/account linkage, simplified sign on, and simple session management. ID-FF is an extension of SAML and served as input for SAML 2.0 (see Figure 2).

The **Liberty Identity Web Services Framework** (ID-WSF) consists of a set of schemata, bindings (SOAP, PAOS), protocols (Discovery and Interaction) and profiles (Security mechanisms, ...) for providing a basic framework of identity services, such as identity service discovery and

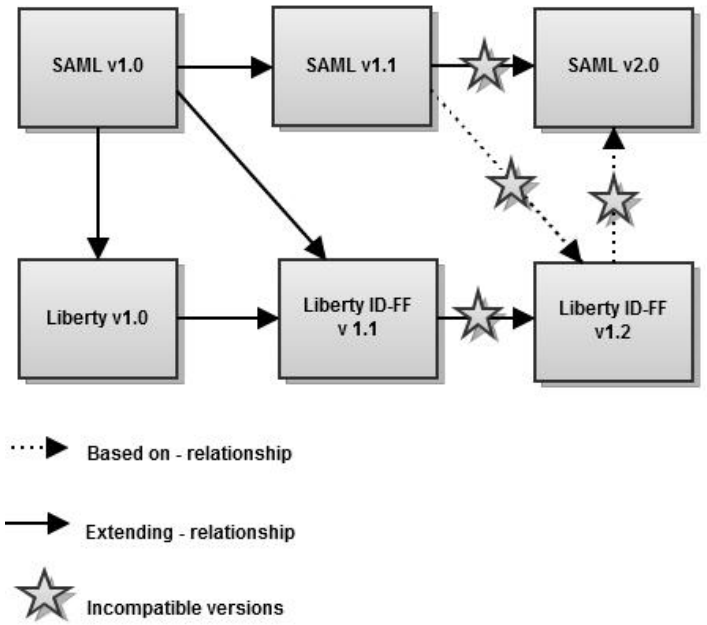

Figure 2: Liberty Alliance ID-FF and SAML

invocation. ID-WSF provides the framework for building interoperable identity services, permission based attribute sharing, identity service description and discovery, and the associated security profiles.

The **Liberty Identity Service Interface Specifications** (ID-SIS) utilize the ID-WSF and ID-FF to provide networked identity services, such as contacts, presence detection or wallet services that depend on networked identity. ID-SIS enables interoperable identity services such as personal identity profile service, alert service, calendar service, wallet service, contacts service, geo-location service, presence service and so on.

Liberty Alliance is currently transitioning to the Kantara Initiative [70]. The Kantara Initiative is working to bridge and harmonize the identity community with actions that will help ensure secure, identity-based, online interactions while preventing misuse of personal information so that networks will become privacy protecting and more natively trustworthy environments.

8.1.3 WS-*

WS-* is a collective noun for a variety of specifications associated with web services. These specifications form together the basic framework for web services build on the first-generation standards of SOAP, WSDL and UDDI. This association does not mean they are developed by a main standard body, the specifications are maintained by a diverse set of bodies or entities. The specifications are also not strictly disjunct , they may complement, overlap, and compete with

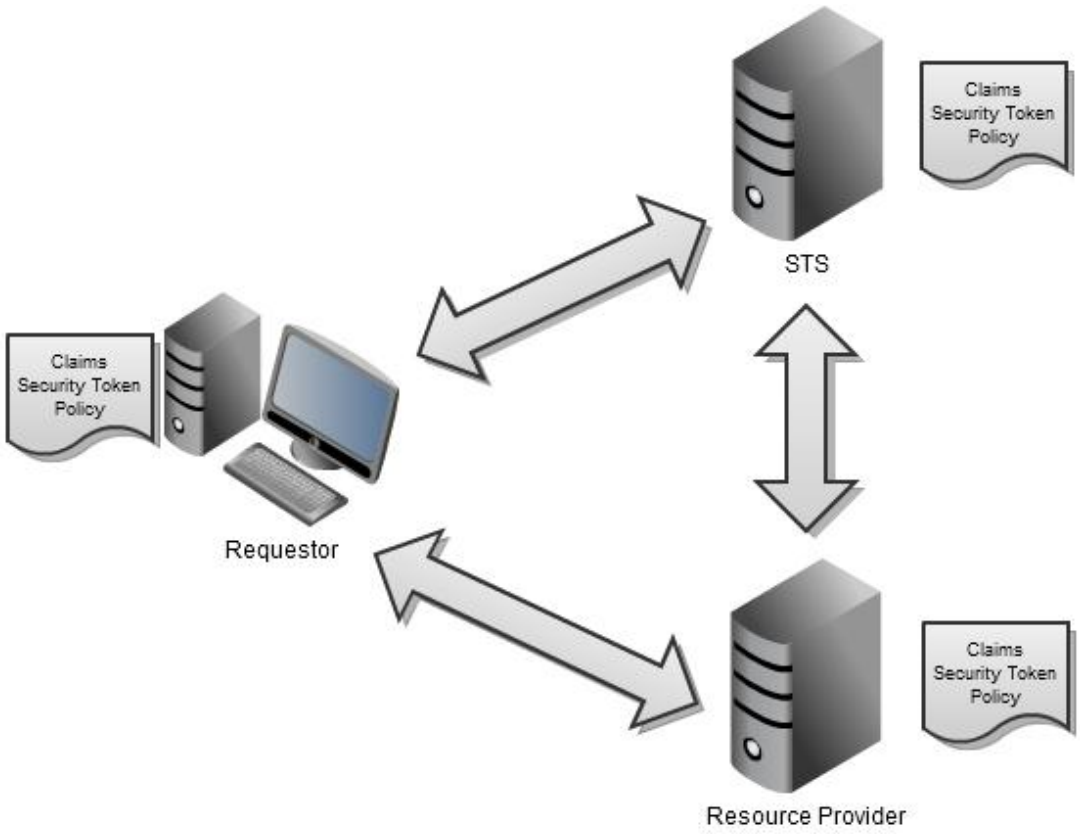

Figure 3: STS Model

each other.

In the domain of security and especially identity management, five specifications of the WS-* are important: WS-Security, WS-Trust, WS-SecureConversation, WS-Federation and WS-SecurePolicy.

WS-Security [71] defines how web service messages can be exchanged in a secure way by guarding the integrity, confidentiality and the sender's identity of the messages. To enforce this, WS-Security uses XML signature (for integrity), XML encryption (for confidentiality) and various security token formats, like SAML, Kerberos, X.509 (for sender authentication), to provide end-to-end security. The type of signature format, encryption algorithm and token model used is the responsibility of the the person who implements WS-Security. This specification does not provide a full security solution for web services, it is just a building block. When implemented in collaboration with other WS-* security specifications, a secure system can be provided (meaning a high responsibility for the person who implements the specifications).

WS-Trust [72] is an extension of WS-Security providing methods for issuing, renewing and validating security tokens and providing ways to establish, assess the presence of, and broker trust relationships. Using the extensions defined in WS-Trust, applications can participate in

secure communication designed to work within the web service framework. A main concept in WS-Trust is the Security Token Service (STS), this is a special web service that issues security tokens conforming to the WS-Security specification. In Figure 3, each arrow represents a possible communication path between the participants. Each participant has its own policies which combine to determine the security tokens and associated claims required to communicate along a particular path.

WS-SecureConversation [73] defines extensions that include session key derivation and security context establishment/sharing. This allows contexts to be established and potentially more efficient keys or new key material to be exchanged, thereby increasing the overall performance and security of the subsequent exchanges.

WS-SecurePolicy [74] extends the WS-Security, WS-Trust and WS-SecureConversation security protocols by offering a specification for policy-based web services .WS-SecurePolicy defines XML based policies that are called security policy assertions. These policies allow web services to express their constraints and requirements. Policies can be used to drive development tools to generate code with certain capabilities, or may be used at runtime to negotiate the security aspects of web service communication. The intent is to provide enough information for compatibility and interoperability to be determined by web service participants.

WS-Federation [75] defines federation mechanisms by extending WS-Trust, WS-Security and WS-SecurePolicy. WS-federation has two profiles. The passive requestor profiles enables browser-based SSO and identity federation. The active requestor profile enables identity federation for web services. A fundamental goal of WS-Federation is to simplify the development of federated services through cross-realm communication and management of federation services by re-using the WS-Trust Security Token Service model and protocol.

8.1.4 OpenID

OpenID [76] is a lightweight HTTP-based protocol for single sign-on and attribute exchange. The OpenID specification is widely adapted and implemented by internet companies that have large user bases like Google, Yahoo, WordPress and Facebook.

OpenID Authentication uses only standard HTTP(S) requests and responses, so no special capabilities are required from the user client. OpenID is not tied to the use of cookies or any other specific mechanism of consumer or OpenID Provider session management. Extensions to user clients can simplify the end user interaction but are not required by the protocol. The OpenID Authentication protocol messages are a defined fixed set of key-value pairs which are included in the HTTP(S) requests as HTTP parameters.

The main OpenID authentication component has some extensions:

- OpenID Simple Registration [77]: allows very light-weight profile exchange. It is designed to pass eight commonly requested pieces of information when an end-user goes to register a new account with a web service.

- OpenID Attribute Exchange [78]: defines how to exchange identity information between endpoints through attributes.
- OpenID Authentication Policy [79]allows a consumer to request that particular authentication policies are applied by the OpenID Provider when authenticating an end user.

The main component in the OpenID architecture is the OpenID provider server. On this server a user can register and receive a personal identifier, this is a string in either URL or XRI form. When authenticating in OpenID, the user presents this identifier to the web application (consumer) he/she wishes to authenticate on. The origin of the identifier that is presented to the consumer is completely the choice of the user, it can reference to any OpenID Provider. The consumer will use the presented identifier to discover the referenced OpenID provider and corresponding authentication endpoint. After the discovery step, the consumer and provider can optionally establish an association by exchanging a shared secret. This removes the need of verification messages for subsequent authentication requests and responses. Subsequently the user is redirected to the provider, where an authentication step takes place. How this authentication occurs is out of scope of the specification, leaving the possibility to skip the authentication step. The user is finally redirected back to the consumer with either a positive or negative assertion. The consumer verifies the signed assertions either through the established shared key (in the optional association step) or by sending a verification message to the provider.

OpenID does not define explicit trust between the OpenID authentication provider and the consumer. OpenID only ensures identification, that the given person is the one he claimed to be during his previous visit. It enables a consumer to reidentify the user but gives no guarantee about the user itself because of the lack trust between consumer and provider. Anyone can use any provider to identify himself or even set up his/her own one. If trust is required, a consumer could require the user to use a specific provider, i.e. Google.

8.1.5 PKIX

PKIX is a standard, specified by the IETF's Public-Key Infrastructure working group, describing a public key infrastructure. It specifies public key certificates, certificate revocation lists, attribute certificates, and a certification path validation algorithm. PKIX is a derivation of the X.509 [80] standard in order to adapt it to the more specific domain of internet standards. The term X.509 certificate usually refers to the IETF's PKIX Certificate and CRL Profile of the X.509 v3 certificate standard.

X.509 public key certificates bind an identity to a public key. It identifies the holder and can be used to authenticate him. Public key certificates typically last for a long time. Authorisation information can be placed in a public key certificate through extensions. It is usually not desirable though to place authorisation information in a public key certificate. Authorisation information typicially has a shorten lifetime and the authority issuing public key certificates usually isn't authoritative for the authorisation information. Authorisation information can be separated by

putting it in an attribute certificate. An attribute certificate is a digitally signed identity with a set of attributes. It doens't have a public key and typically doesn't last as long as a public key certificate. PKIX is mainly used for authentication though and less commonly for authorization purposes (RFC 3281 [81]).

The main building blocks in the PKIX system are Central authorities (CA's), these are entities that can issue certificates. Each of these entities contains a signed certificate(s) with public/private keypair. A CA signs every new issued certificate with his private key and includes a reference to his own certificate. In this way every certificate can be verified by using the public key of the CA that issued it. An issued certificate signed by a CA can also be used as base certificate of a new CA. It is clear that in this way a structured system can be build of CA's and end entities (base certificates that are not used as base for a CA). These structures can range from very simple, hierarchical topologies to complex topologies such as mesh architectures.

The certification path validation algorithm [82] is the algorithm which verifies that a given certificate path is valid under a given PKI. A path starts with the end entity and proceeds through a number of intermediate certificates up to a trusted root certificate, typically issued by a trusted certification authority. Path validation is necessary for a relying party to make an informed trust decision when presented with any certificate that is not already explicitly trusted.

The revocation of certificates is specified in the Certificate Revocation List (CRL) profile. The main concept behind CRL is the generation of a blacklist containing a list of all revoked certificates. When the CRL is consulted, the whole list of revoked certificates is returned. An alternative for CRL is the Online Certificate Status Protocol (OCSP), here a request to the list of revoked certificates returns a signed response signifying that the certificate specified in the request is *good*, *revoked*, *unknown* or in *error*. The OCSP can be seen as a lightweight pseudo-CRL that avoids CRL network bandwidth problems.

Although PKIX certificates are widely spread there are some issues. As explained before PKIX is a derivation of X.509, a standard from 1988, some of the design decisions are not compatible with the current widely accepted computer (internet) standards. The CRL system for example is not a good design solution to revoke certificates, it slows down the overall performance of the PKI infrastructure. Because of this many of the X.509 implementations turn off the revocation checks. Another example is that the X.509 specification is over-functional and underspecified and the normative information is spread across many documents from different standardization bodies.

8.1.6 U-Prove

U-Prove is a claims-based identity management framework that aims to offer anonymity, security, scalability and privacy based on cryptographic technologies. The end-user in the U-Prove system is offered extensive control over his/her personal information. He/she can decide what information he/she wants to share with a service provider in order to make a claim. For example if a user wants to buy a song on Amazon, he/she shares all his personal account information (like address, ...) that is stored in the Amazon database (generated when he/she created an account). To handle the purchase, Amazon does not need to know all this personal information of the user.

Only the credit card number is sufficient to make a claim (downloading a song does not need for example an address). With U-Prove, the user can decide to publish only his credit card number to make a claim for purchasing the song while other sensible information is kept hidden.

At the heart of the U-Prove technology is the notion of a U-Prove token.This token is used as base for authentication of an end-user at a service provider and is issued by a third party identity provider. The content of a digital signed U-Prove token is a public key together with a collection of attributes that represents the personal information that the end-user permits the service provider to see.

The main authentication profile of U-Prove starts with an end-user that browses to a service provider. The service provider sends the end-user to a trusted identity provider together with a list of attributes that are needed to make a claim. The end-user approves the list of attributes that are requested and a 3-leg interactive protocol is started to generate a new U-Prove token authenticating the end-user. In this protocol the user gives his/her credentials, the list of attributes that is needed by the service provider and a random generated public key (the corresponding private key is kept by the end-user) to the identity provider. The identity provider receives the requested attributes from his back-end and generates a U-Prove token. If the token is generated successful the end-user is redirected back to the service provider where a presentation protocol is initiated. In this protocol the U-Prove token is verified (using the available public key of the identity provider) and a challenge (containing a nounce) is send from the service provider to the end-user. That challenged is then encrypted with the end-user's random private key and send back to the service provider. In this way the end-user can be verified. If the previous steps succeeded, the service provider can check the claim and make an access decision.

U-Prove is very interesting from the point of end-users. Internet privacy has become a hot issue last years, this mainly because of the increase of internet crime (stealing confidential information). Giving an end-user more control over his/her private information can be one of the solutions to halt this. For websites the system is less interesting. The main problem with the system is that it limits the data mining capabilities of servers that have installed U-Prove.This is a big problem for sites like Amazon who mainly work on clever data mining techniques to advertise to their user base. This makes U-Prove only suitable for a small segment of the internet market.

8.1.7 Shibboleth

Shibboleth is an architecture and implementation of a federated single sign-on authentication and authorisation infrastructure heavily coupled with SAML. It was realized by the Internet 2 networking consortium. The primary function of the Shibboleth system is to support identity federation between multiple sites using the SAML protocol standard. Shibboleth's added value lies in support for privacy, business process improvement via user attributes, extensive policy controls, and large-scale federation support via metadata. Hence Shibboleth accommodates richer and more complex metadata distributed by a federated operator. It has more refined capabilities for managing trust implicit in larger communities. It allows users and enterprises to manage attribute releases, reflecting the greater number and variety of participants.

The Shibboleth and SAML design processes have always been coupled to ensure that Shibboleth is standards-based. It uses SAML formats and protocol bindings whenever possible and appropriate. The first version of Shibboleth was released in 2003 based on SAML version 1.0 (some of the Shibboleth principals were authors in SAML 1.0). In 2005 version 1.3 was released based this time on the new specifications contained in SAML version 1.1 (and the previous Shibboleth versions). This version (1.3) was in his turn used to specify the new SAML 2.0 procotol. Currently Shibboleth 2.0 is the last released version, depending heavily on the SAML 2.0 specifications.

The Identity Provider (IdP) and Service Provider (SP) are the main architectural components in Shibboleth. The identity provider is an entity that authenticates principals and produces assertions of authentication and attribute information in accordance with the SAML Assertions and Protocols specification and the SAML browser profiles in the SAML Bindings and Profiles specification. A service provider is an entity that provides a web-based service, application or resource subject to authorisation or customization on the basis of a security context established by means of a SAML browser profile. Next to these two components there is also an optional Discovery Service component. This component can keep track (in case of multiple IdP's) of the IdP that was selected by a user, using a browser cookie (specified in the Identity Provider Discovery Profile). However this can only store the fact that an IdP was selected, not whether the user authenticated successfully (in which case the user might wish to change their choice of IdP). Hence, by default the discovery service does not automatically redirect the request to the last selected IdP, instead a list of the most recently selected IdPs is provided as a hint.

The Shibboleth profiles and protocols are again heavily based on SAML. The most important are: Authentication Request Profile, Browser/POST Authentication Response Profile, Browser/Artifact Authentication Response Profile, Attribute Exchange Profile, Transient NameIdentifier Format and the Metadata Profile. An important SAML profile that is not implemented in Shibboleth is the Single Logout profile. This mainly because there is no clean solution to implement this profile properly.

The implementation part of Shibboleth offers an implementation of the three main components meeting the profile and protocol requirements. It is currently focused on web browser applications, meaning that there is limited support for web services standards like the WS-* specifications.

8.1.8 CAS

CAS [83] is a centralized ticked based single sign-on HTTP-based protocol. Unlike most of the single sign-on systems, CAS lacks the use of attributes.

When a user visits a website requiring CAS authentication, the site redirects the user to a CAS server. This is the central component in the CAS architecture and is responsible for authenticating users, issuing security tickets and validating these tickets. Once the user has successfully authenticated himself, the CAS server redirects the user back to the website passing along with a security ticket. The website validates this security ticket bye contacting the CAS server. If the

validation succeeds, the user is successfully authenticated on the website.

CAS 2.0 added support for proxiable credentials which allows a user to be authenticated on back-end services of a website through proxy tickets. In the proxy scenario, the user has authenticated him/herself on a portal website that contains back-end services. Now the user wants to authenticate him/herself on one or more of the back-end services. For this, the portal website requests a proxy ticket from the CAS server, that contains the needed information to authenticate the user to the back-end service. Next the portal website send this proxy ticket to the back-end service. The back-end service sends on his turn the proxy ticket to the CAS server for validation. If the validation succeeded the user authenticated him/herself to the back-end service.

8.1.9 Kerberos

Kerberos [84] can be seen as the first widely distributed single sign-on system. It is a ticket oriented system that allows a user to authenticate him/herself in a non-secure network domain. The authentication is mutual, both the user and the server verify each other's identity. It is based on symmetric key cryptography and requires a trusted third party. Kerberos offers sufficient protection facilities against eavesdropping and replay attacks.The user credentials are not communicated over the untrusted network. The Kerberos system demands that the users are trusted else the Kerberos system is at risk of being compromised.

8.2 Authorisation

8.2.1 OAUTH

OAuth [85] is an open standard for authorisation. It allows a resource owner to grant access to his/her private resources on one site (which is called the server), to another site (called client) without the need to share his/her personal credentials. Instead OAuth uses shared tokens which can be limited in time and/or scope. As such the resource owner can give different clients different levels of access. The OAuth protocol messages are a defined fixed set of key-value pairs which are included in the HTTP(S) requests as HTTP parameters.

Three kinds of credentials are used during the OAuth workflow.

1. A client credential is used to authenticate a client to the server.
2. A token credential authorises a client to access a owner's resource limited in time and/or scope.
3. A temporary credential is a shared secret that identifies an authorisation request.

A client that wants to access a particular resource on a server needs a client and a token credential. The client credential can be obtained when the client registers him/herself at the

server by providing his/her credentials. How the token credential is generated, is explained in the following steps. The resource owner browses to the client (containing client credentials issued by the server where the resources are situated) and requests the client to access his/her server resource. The client obtains a request token and shared secret (temporary credential) from the server. The request token, which represents the not yet authorized request of the client, is returned by the server. After the token is received the request token needs to be authorized, therefore the resource owner is redirected to the server where he authenticates him/herself. If the authentication is successful the resource owner is explicitly prompted to grant the client access to the protected server resources. If access is granted the resource owner is redirected back again to the client. In the last step the client uses the authorized request token to obtain a new shared secret and access token (token credentials) from the server.

8.2.2 Attribute-Based Authorisation

In an attribute-based authorisation model, identity information on a subject is exchanged from one site to another site in support of some action. The identity information exchanged will rather be information on some characteristic of the subject relevant to the action performed (i.e. the role of the subject in a given scenario), opposed to information exchanged during authentication on who, when and how the subject authenticated. To exchange this authorisation information the same languages can be used as for exchanging authentication attributes (i.e. SAML, WS-*, Liberty, X.509).

8.2.3 XACML

XACML [86] (full name: eXtensible Access Control Markup Language) is a XML based declarative access control policy language defining both a policy, decision request and decision response language. It is based on the Attribute Based Access Control (ABAC) model which incorporates Role Based Access Control (RBAC). Currently version 2.0 of XACML is adopted, the planned 3.0 version (which is currently in first draft) will support a broad range of new features. In the following paragraphs version 2.0 is used as reference unless otherwise stated.

Central in the XACML architecture stand two main components: the Policy Enforcement Point (PEP) and the Policy Decision Point (PDP). The PEP intercepts access requests by a subject on one or more protected resources on a server. After the PEP has intercepted such an access request, it generates a decision request based on the attributes of the subject, the resource in question, the performed action and other information pertaining to the request. This decision request is send to the PDP, a component responsible for making access decisions. For making an access decision the PDP interprets the incoming decision request and searches for policies (known to the PDP) that apply to the request. The search for matching policies can be an overhead in the XACML system, because for each decision request, every policy available to the system is checked (no indexing mechanism available). After an access decision is made, based on the access rules in the policies, the PDP generates a decision response and sends it back to the PEP. The PEP uses this response to

decide if access is granted to the protected resource or not.

The XACML policy language [87] is designed to offer great interoperability with different platforms and extensibility. The smallest element in the language is the rule element. Each rule targets a set of decision requests to which it is intended to apply. The rule target is expressed in form of a logical expression on attributes of the request. A rule also has a condition element that refines even more the set of decision requests that are targeted. If for the given target and condition a set of decision requests is found, the rule responses with a defined effect value that permits or denies access to the selected set of decision requests. Rules can not exists on there own, they have to be combined in a policy element. This policy element groups rules and combines the access decisions (permit or deny), that were evaluated for the targeted set of decision requests, to a general policy access decision using a combining algorithm. A policy element can target a set of decision requests. Finally there is an optional third element called policyset. This element can group policy elements and/or other policysets. Like the policy element it has a combining algorithm to generate a general policyset access decision. The policyset can target a set of decision requests. Policies and Policysets can contain obligations, which contain a set of operations that must be performed by the PEP in conjunction with an authorisation decision

XACML 3.0 adds a new important concept to the policy language: delegation. It permits some users to create policies of limited duration to delegate certain capabilities to others.

8.2.4 Ponder

Ponder [88] is a declarative, object-oriented language for specifying different types of policies, grouping policies into roles and relationships and finally defining configurations of roles and relationships as management structures. Ponder can be used to specify security policies with role-based access control, as well as general-purpose management policies. Ponder2 [89] is the successor of the Ponder policy language. It corrects some of the main disadvantages of Ponder making the framework easier to deploy, make it more accessible for small devices and improve the collaboration/interaction/federation between the policy execution components.

Key concepts of the language include:

- Domains, which provide a means of grouping resources to which policies apply and can be used to partition the resources in a large system according to geographical boundaries, resource type, responsibility and authority or for the convenience of human managers. Membership of a domain is explicit and not defined in terms of a predicate on resource attributes. A domain does not encapsulate the resources it contains but merely holds references to resource interfaces.
- Roles, which group the policies relating to a position in an organisation (like service administrator, service operator, ...)
- Relationships, which define interactions between roles and management structures to define a configuration of roles and relationships pertaining to an organisational unit such as

a department.

Ponder supports, as said before, an extensible range of policy types.

- Authorisation policies are essentially security policies related to access-control and specify what activities a subject is permitted or forbidden to do to a set of target resources.
- Obligation policies specify what activities a subject must do to a set of target resources and define the duties of the policy subject. Obligation policies are triggered by events and are normally interpreted by a manager agent at the subject.
- Refrain policies specify what a subject must refrain from doing and are similar to negative authorisation policies but are interpreted by the subject.
- Delegation policies specify which actions subjects are allowed to delegate to others. A delegation policy thus specifies an authorisation to delegate.
- Composite policies are used to group a set of related policy specifications within a syntactic scope with shared declarations in order to simplify the policy specification task for large distributed systems. Four types of composite policies are provided: groups, roles, relationships and management structures.
- Constraints can be specified to limit the applicability of policies based on time or values of the attributes of the resources to which the policy refers.
- Meta-policies are policies about which policies can coexist in the system or what are permitted attribute values for a valid policy.

8.2.5 PERMIS

PERMIS [90] (PrivilEge and Role Management Infrastructure Standards) is an authorisation infrastructure that is based on two underlying technologies: role based access control (RBAC) and Policy based Management. RBAC allows to group all users who use the infrastructure into roles (or attributes). Each role (or attribute) is associated with a collection of privileges. A user's membership of a role will allow the user to exercise the privileges associated with the role. The policy based management is a collection of rules that specify the authorisation criteria for the users.

Two management services are provided in PERMIS: privilege management and policy management. The privilege management service provides services for allowing managers to allocate privileges to users. This generated privilege information is stored in X.509 Attribute Certificate format. The privilege management service also provides a service, which allows users to delegate (a subset of) their privileges to other users in their domain. In the policy management component, PERMIS provides a policy editor to allow administrators to easily construct authorisation policies for their application and delegation policies. The policies are created in

XML format, and may then be optionally protected by encapsulating in an X.509 policy attribute certificate, digitally signed by an administrator. This offers an extra service to the policies by providing integrity protection and tampering detection.

For authorisation decision making, PERMIS provides two main components: the credential validation service (CVS) and the policy decision point (PDP). The CVS is used to validate if the allocation of privileges is valid or not. (The need for this is due to the fact that privileges may be managed in a distributed manner, thus potentially anybody can allocate any privileges to anyone else, but only some of these allocation will be recognised by the PERMIS CVS as being valid). After the validation step, the CVS returns a set of valid attributes for a user, ready for the PDP to make an authorisation decision. The PDP renders an authorisation decision for a user's access request, normally in the form of granted or denied. The application that has send the request is responsible for enforcing the decisions returned from the PDP.

8.2.6 GAS

The Gridge Toolkit [91] is a set of integrated middleware services, based on GridLab [92], used to build grid environments. One of the services provided by Gridge is the Gridge Authorization Service (GAS). GAS is an authorisation system that act as the standard decision point for the components in a grid-based system. Its main goal is to fulfil the authorisation requirements for these components. It holds security policies (for all the connected grid components) which are used to make authorisation decisions upon client requests. GAS has been designed in such a way that it can be integrated with external components easily and can manage security policies for complex systems. GAS has a flexible modular structure, making it possible to extend the GAS functionality with new communication modules. This modules can be used to model many abstract and real world objects and security policies for such objects.

8.2.7 Cassandra

Cassandra [93] is a role-based policy specification for access control in a distributed system in which the expressiveness (and the computational complexity) can be tuned according to need by choosing an appropriate constraint domain. It is designed to satisfy complex policy requirements and at the same time be simple enough to express access control semantics that can be defined in a formal way (trade-off between expressiveness and decidability). This means that, depending on the application's requirements, a suitable constraint domain can be used in the Cassandra system without the need of changing the semantics.

The language is based on a clear mathematical foundation, more precisely Datalog with constraints (subset of Prolog) and uses five special predicates to express a wide range of policies including role hierarchy, role delegation, separation of duties, cascading revocation, automatic credential discovery and trust negotiation. It is policy neutral meaning it can express subtle variants of well-known policy idioms. The formal specification for Cassandra does not only include policy language semantics but also operational semantics. A goal-oriented distributed

policy evaluation algorithm is used to guarantee termination.

8.3 Data storage security

8.3.1 IEEE Standard for Encrypted Storage

The Institute of Electrical and Electronics Engineers (IEEE) approved in August 2002 **Project 1619** (IEEE P1619) to develop standards for the protection of information on storage media, including encryption and key management. The Security In Storage Working Group (SISWG) was assembled, by the IEEE, to oversee the work on those standards.

The first approved IEEE P1619 standard is the **Standard for Cryptographic Protection of Data on Block-Oriented Storage Devices** (IEEE P1619-2007, a.k.a. P1619 Narrow-Block Encryption [94]). The standard describes the methods, algorithms and modes of data protection to be used for cryptographic protection of stored data on sector-based devices, including import/export methods of the keys, which enables interoperability between different implementations. Major results of the work in P1619-2007 are:

1. A standard for XML-based key-export format
2. A standard for the XTS-AES (**A**dvanced **E**ncryption **S**tandard) encryption algorithm, an **X**EX-based (XOR-Encrypt-XOR) Tweaked CodeBook (**T**CB) with CipherText Stealing (CT**S**) algorithm, that is suitable for encryption of stored data in fixed-block devices. Thanks to CTS the block size used by the device did not need to be divisible by the cipher block size.

However, the importance of making a distinction between narrow-block and wide-block encryption was made to tackle the problem of malleable encryption. Narrow-block encryption has the advantage of more efficient hardware implementation, but the smaller block size provides a finer granularity for data modifications attacks. SISWG selected a narrow-block (16 bytes or128 bits) encryption with no authentication in the P1619 standard. For this, the **Standard for Authenticated Encryption with Length Expansion Storage Devices** (IEEE P1619.1) specifies methods for guarantying the privacy and integrity of stored data, while the wide-block encryption is tackled in the **Standard for Wide-Block Encryption for shared Storage Media** (IEEE P1619.2).

The IEEE P1619.1 (Authenticated Encryption) standard specifies encryption and data authentication procedures for storage devices that support length expansion (e.g. tape drives). All these procedures are cryptographic modes of operation using the NIST-approved AES-256 block-cipher:

- CCM-128-AES-256: **C**ounter mode encryption with **C**BC\-**M**AC (**C**ipher **B**lock **C**haining **M**essage **A**uthentication **C**ode)
- GCM-128-AES-256: **G**alois/**C**ounter **m**ode (counter mode encryption with 128-bit finite field message authentication code)

- CBC-AES-256-HMAC-SHA: cipher block chaining mode encryption with key-hash message authentication code using secure hashing algorithm
- XTS-AES-256-HMAC-SHA: XTS encryption with hash-key message authentication code using secure hashing algorithm

The IEEE P1619.2 (Wide-block Encryption) standard describes in its latest draft (see status of P1619.2 [95]) the EME2-AES and XCB-AES wide-block encryption modes to meet its scope of specifying the architecture for encryption of data in random access storage devices, oriented towards applications which benefit from wide encryption-block sizes of 512 bytes and above.

The initial goal towards key management was envisioned for the **Standard for Key Management Infrastructure for Cryptographic Protection of Stored Data** (IEEE P1619.3), however as of December 2010, the Security in Storage Working group has voted to remove this PAR (Project Authorization Request), due largely to the approval of the OASIS KMIP [96] (**K**ey **M**anagement **I**nteroperability **P**rotocol) specification (see SISWG note [97]).

8.4 Data transfer security

8.4.1 TLS/SSL

A reliable protocol for securing data transfer is SSL [98] together with his successor TLS [99]. They both offer services to secure the packets of the application layer of the OSI-model [100]. These services provide authentication, message integrity and confidentiality over an insecure network. TLS (SSL) is commonly used in the HTTPS protocol and for tunnelling in VPN.

There is no big difference in functionality between SSL (v 3.0) and TLS (v 1.0), but both standards are not compatible. TLS though contains a means that makes it possible to downgrade TLS to SSL. The main changes in TLS are fixes for security holes and the enhancement of the client and server hash/encryption algorithms.

To set up a secure connection the protocol uses several steps called the handshake procedure. First the server and client that want to exchange sensitive data over TLS (SSL) negotiate the cipher suite (this are the cypher and hash algorithms) that will be used by both parties as base for encryption and hashing of the data. After this step the server sends a digital certificate containing the server name, the CA that signed the certificate (to verify the certificate can be trusted) and the public encryption key of the server. After the client has checked the validity of the certificate, the client encrypts a random number with the public key of the server and sends it to the server (which can only be decrypted by the server's private key). Finally both client and server generate session keys, using the random number as base, that will be used for the secured connection. The client and server can now exchange the sensitive data in a secure way.

Because the public key operations in the handshake procedure are computationally expensive, the TLS protocol offers a method to avoid these consuming operations (called the resumed TLS handshake). When initializing the handshake procedure the server generates a session id and

send this to the client. The client associates this id with the ip and port of the server, while the server associates this id with the cryptographic parameters previously negotiated. When the client connects again to this server, it uses the session id to skip some parts of the CPU consuming steps in the handshake procedure.

8.4.2 IPSec

Next to the well-known SSL-TLS standards there is an other protocol that is specified for securing data that needs to be transmitted. IPSec (Internet Protocol Security) [101] secures the packets of the network layer of the OSI-model [102] by encrypting and/or authenticating the IP packages of this layer. This allows network applications to communicate in a secure way on a network without the need of implementing extra network software in their application stack. This application independence is not possible in TLS-SSL systems. The drawback of this, is that IPSec is more complex to deploy than TLS-SSL.

The main goals for transmitting data in IPSec are: confidentiality, integrity and authentication. For this IPSec provides two main kinds of services: encapsulating security payload (ESP) and Authentication Header (AH). ESP provides confidentiality, authenticity and connectionless integrity protection for the communication and provides protection against replay attacks (by using the sliding window technique and discarding old packets). AH ensures protection of the authenticity and connectionless integrity of the data and provides protection against replay attacks (by using the sliding window technique and discarding old packets). ESP and AH can be used together or separately in an IP packet (they operate directly on top of IP).

IPSec has two main modes of operation:

- Transport mode: in this mode only the payload (the data you transfer) of the IP packet is encrypted and/or authenticated.
- Tunnel model: here the entire IP packet is encrypted and/or authenticated. It is then encapsulated into a completely new IP packet containing a new IP header. This operation is commonly used in VPN.

The encryption of small packages in the network generates a large computational overhead which can be a problem in real-time systems.

9 Interoperability standards

Relevant to WP7, WP8, WP11, WP12, WP13

9.1 Interoperability in healthcare

The report "Semantic Interoperability for Better Health and Safer Healthcare" [103] produced by the FP6 SemanticHEALTH project provides a number of relevant definitions, standards, and application domains for semantic interoperability. SemanticHEALTH developed a longer-term research and deployment roadmap for semantic interoperability. Its vision is to identify key steps towards realising semantic interoperability across the whole health value system, thereby focusing on the needs of patient care, biomedical and clinical research as well as of public health through the re-use of primary health data.

The Recommendation, COM(2008)3282 [104] provides the following definition: "Semantic interoperability means ensuring that the precise meaning of exchanged information is understandable by any other system or application not initially developed for this purpose", whereas "interoperability of electronic health record systems means the ability of two or more electronic health record systems to exchange both computer interpretable data and human interpretable information and knowledge", (page 14).

In the context defined above, *semantic interoperability* (SIOp) addresses issues of how to best facilitate the coding, transmission and use of meaning across seamless health services, between providers, patients, citizens and authorities, research and training. Its geographic scope ranges from local interoperability (within, for example, hospitals or hospital networks) to regional, national and crossborder interoperability. The information transferred may be at the level of individual patients, but also aggregated information for quality assurance, policy, remuneration, or research.

Interoperability is split according to the SemanticHealth report in following 4 incremental levels:

- Level 0: no interoperability at all
- Level 1: technical and syntactical interoperability (no semantic interoperability)
- Level 2: two orthogonal levels of partial semantic interoperability
- Level 2a: unidirectional semantic interoperability
- Level 2b: bidirectional semantic interoperability of meaningful fragments
- Level 3: full semantic interoperability, sharable context, seamless cooperability

The investigations undertaken by SemanticHEALTH suggest that full semantic interoperability (Level 3) is required in order to take full advantage of computerized medical records. It is however also recognized that due to steep investments needed, the highest level of semantic interoperability should only be sought in specific areas with the high potential for improvements, while in other areas a lower interoperability level may suffice.

In order to achieve complete technical and semantic interoperability, existing standards have to be harmonized and bridged. In the US major consortia have been formed to provide semantic

interoperability between the standards (e.g. BRIDG covering CDISC, HL7 (RCRIM), FDA, NCI) and to provide core sets of data collection fields (CDISC CDASH). Furthermore, efforts exploring the potential to improve interoperability between the electronic health record and electronic data capture (e.g. CDISC eSDI, eClinical Forum/PhRMA EDC/eSource Taskforce) have been initiated.

9.2 Generic protocols for web services and network communication

The World Wide Web Consortium (W3C) produces protocols and standards for Web technologies.

OWL-based Web Service Ontology (OWL-S) is an OWL based Ontology, within the OWL-based framework of the Semantic Web, for describing Web services. OWL-S ontology is also sometime considered as a language for describing services, reflecting the fact that it provides a standard vocabulary that can be used together with the other aspects of the OWL description language to create service descriptions

Web Service Modeling Ontology (WSMO) is an ontology for semantically describing Semantic Web Services. It is a model for the description of semantic web services that tries to overcome the limit of the existing technologies for the service description, in particular OWL-S. Web Service Modeling Language (WSML) is a language that formalizes the WSMO. It uses well-known logic formalisms, namely, Description Logics, First-Order Logic and Logic Programming, in order to enable the description of various aspects related to Semantic Web Services.

Web Service Semantics - WSDL-S specification is a W3C Member Submission that defines how to add semantic information to Web Services Description Language (WSDL) documents. WSDL is a XML based language that is used to describe the functionalities of a web service. WSDL-S tries to overcome the lack of semantics in WSDL by adding new extensibility elements to the WSDL standards to annotate the semantic of web services. Each service description refers one or more Semantic Model. A semantic model captures the terms and concepts used to describe and represent an area of knowledge or some part of the world, including a software system. A semantic model usually includes concepts in the domain of interest, relationships among them, their properties, and their values. WSDL-S provide mechanisms to annotate the service and its inputs, outputs and operations. Additionally, it provides mechanisms to specify and annotate preconditions and effects of Web Services. These preconditions and effects together with the semantic annotations of inputs and outputs can enable automation of the process of service discovery.

Semantic Annotation for WSDL (SAWSDL) defines how to add semantic annotations to various parts of a WSDL document such as input and output message structures, interfaces and operations. The extension attributes defined in this specification fit within the WSDL 2.0 extensibility framework. It provides mechanisms by which concepts from the semantic models that are defined either within or outside the WSDL document can be referenced from within WSDL components as annotations.The annotations on schema types can be used during Web service discovery and composition. In addition, SAWSDL defines an annotation mechanism for specifying the data mapping of XML Schema types to and from an ontology; such mappings could be used during invocation, particularly when mediation is required.

9.3 Information Extraction Tools

One of the main techniques of information extraction applied to medical documents is classification. Classification of documents based on free-text can be performed using text categorization methods from Information retrieval or machine learning techniques. Efforts have been made to automatically classify the radiology reports based on their content. Wilcox et al [105], assigned 6 different clinical conditions to narrative x-ray reports using different machine learning techniques. Patients' cancer stage has been inferred by classification of histology reports using statistical machine learning methods. [106] Comparison of IR, ML and rule based approaches for ICD-9 CM code assignment to radiology reports carried out by Goldstein et al. [107] Also investigated is the role of domain knowledge in automatic classification of medical reports [108].

In addition to classifying the medical documents as a whole, machine learning techniques has been applied to classify a part of the documents like the Medical Abstracts sentences [109] or the assertions of findings in medical reports [110].The effect of feature representation on classification of medical documents has been investigated by representing the document in Bag of Words and Bag of Phrase format [111].

An example of a common information extraction tool for biomedical domain is MedLEE. It is a natural language processing tool for radiology reports which uses a medical vocabulary for mapping words and phrases to standard medical terms while also adding some modifier information like body location, region, certainty, etc. to them. It works based on a sentence at a time and does not capture the context.

MedLEE is in routine use for encoding radiology reports, and its ability to identify medical findings in radiology reports has been investigated in several studies [112]. Also,the encoded output of Medlee has been used in many studies. For example, Medlee was used to encode radiology reports for creating feature vectors [105] or structured output generated by Medlee consisting of findings and modifiers has been used to automatically assign Unified Medical Language System (UMLS) codes to a clinical document [113].

9.4 Clinical Decision Support Tools

In Kawamoto et al [114], efforts and issues concerning use of standards in the CDS area to enable interoperability are discussed. The paper argues that a critical need for enabling CDS capabilities on a much larger scale is the development and adoption of standards that enable current and emerging CDS resources to be more effectively leveraged across multiple applications and care settings. Standards required for such effective scaling of CDS include:

1. Standard terminologies and information models to represent and communicate about health care data
2. Standard approaches to representing clinical knowledge in both human-readable and machine-executable formats

3. Standard approaches for leveraging these knowledge resources to provide CDS capabilities across various applications and care settings.

Item 1 above has been addressed in other sections of this report and is not specific to CDS, but refers to more general requirements for clinical data representation and exchange and for semantic interoperability.

Several languages exist which could be applied for representing and sharing clinical knowledge in a formalized way (executable clinical knowledge according to the above classification), such as Arden syntax [115], Gello (M. Sordo et al. Software specifications for gello: An object-oriented query and expression language for clinical decision support), ERGO (S. Tu et al. Ergo: A template based expression language for encoding eligibility criteria). The Arden Syntax is a standardized language used to represent and share clinical knowledge through Medical Logic Modules (MLMs). The Arden Syntax was introduced in 1989 and was first adopted in 1992 by ASTM; it is not a full-fledged programming language, it is meant to be written and maintained by clinicians. Version 2.0 was adopted in 1999 by both its current sponsor , HL7, and ANSI. version 2.1 is the current standard.

The Guideline Elements Model (GEM) is an example of standard for non-executable representation of clinical knowledge. It is an ASTM standard and is used for representation of the contents of clinical practice guidelines in a structured way. GEM II includes and XML Schema that defines a structured format for extracting relevant content out of a clinical practice guideline, an object-oriented data model and an editor for GEM guidelines.

While several, mainly HL7-supported, standardization efforts emerge that address the use of executable clinical knowledge and the CDS delivery, their adoption is low and significant gaps still exist.

9.5 Semantic Web Standards

9.5.1 Resource Description Framework (RDF)

The Resource Description Framework [116] or RDF, is a data model developed by the World Wide Web Consortium (W3C) for representing semi-structured information in the Web. It enables users to describe Web content with arbitrary vocabularies and associate semantics to the data through the definition of metadata. It has since grown into a general data model and is very widely used outside the realm of web resource description. The RDF data model represents information in terms of statements, commonly known as RDF triples. Each statement in the model describes a resource (the triple's subject) by specifying a property (the triple's predicate) and its value (the triple's object). A triple can also be seen as a directed graph with a the subject and object being represented as nodes and the predicate encoded as an arc between the two nodes. A set of statements describing a set of resources can be seen as an RDF graph and can optionally be associated with a name (URI) to uniquely identify them (the so-called Named Graphs). RDF is a model rather than a format: graphs can be serialized using multiple notations. One of the most

common ones is RDF/XML that serializes an RDF graph in XML, described in **RFC 3870**. Other, more efficient and less verbose, notations also exist such as N3, N-Triples and Turtle. All formats have their benefits and are easily supported for both reading and writing. Since its conception RDF has become the *de facto* standard for representing semi-structured information in the Web in machine-processable form and has become the foundation of the Semantic Web. RDF as a data exchange format also enables interoperability between applications.

9.5.2 Resource Description Framework Schema (RDFS)

RDF does not make any assumption as to the vocabulary used for describing resources nor does it assumes or define the semantics of a given domain. In order to do that users need to define their own vocabularies using a vocabulary description language. RDFS [117] is a W3C recommendation of a basic ontology language for defining vocabularies. It enables users to define concepts (classes of resources) and properties that can be used for describing the semantics of a given application domain. Classes and properties defined in RDFS can then be used to describe an application domain with RDF. In addition, RDFS provides a semantics for specicalization/generalization of classes and properties.

9.5.3 Ontology Web Language (OWL)

OWL [118] (also known as OWL 1.0) is a Web ontology language developed by the Web Ontology working group of the W3C. It has been a W3C Recommendation since 2004. It enables users to specify ontologies by defining classes of objects, properties and relations between classes. It also allows for the specification of individual objects in the domain of discourse. OWL has a higher expressiveness than that of RDFS. Three fragments or dialects of OWL exists. The first is OWL-Lite, the least expressive fragment of the three. OWL Lite is meant for users in need of classification hierarchies and simple constraints. The second fragment is OWL-DL, a more expressive language with good computational properties. Finally, OWL-Full is the most expressive fragment of the three although it provides no guarantees with respect to its computational properties. Other varitions of OWL also exist including OWL Horst and OWL DLP. Many ontologies, including biomedical ontologies, are expressed using some variant of OWL.

9.5.4 Ontology Web Language 2 (OWL 2)

OWL 2 [119] is a revision and extension of OWL 1.0. It includes three versions to accommodate the needs of different type of users. OWL 2 EL is a subset of OWL 2 for which the basic reasoning services can be solved in polynomial time. It is particularly useful for expressing ontologies with large number of classes and/or properties. Many biomedical ontologies can be expressed with OWL 2 EL and many efficient reasoning systems for this language have been developed in recent years. OWL-QL is tailored for applications that use large amount of instance data and whose

main reasoning service is query answering. OWL 2 RL is a profile aimed at applications that require scalable reasoning without losing too much expressive power. Reasoning in OWL 2 RL can be implemented using rule-based approaches.

9.5.5 RDF Query Languages (SPARQL)

SPARQL (SPARQL Protocol and RDF Query Language) [120] is a W3C Recommendation of a query language for RDF. It is intended to replicate the functionality of SQL as used in conjunction with a relational database. It is based upon describing multiple templates for subgraphs, set operations upon the sets of matching subgraphs, and selecting which nodes or edges within the matches are returned as results. It is a mature and widely supported querying language, and is being actively extended and refined to replicate some of SQL's more advanced functionality. Since its inception it has become one of the most widely used query languages for semi-structured data expressed in RDF and it is nowadays a Semantic Web standard query language. Other RDF querying languages exist, but they are usually specific to a particular RDF store or interface implementation. Virtually all RDF repositories nowadays provide an implementation of the language and many data repositories in the Web provide query services backed by SPARQL, the so-called SPARQL endpoints.

9.6 Assembly of workflows

A scientific workflow is a series of connected steps that describes a sequence of operations with regards to computational or data related tasks. There exists a number of scientific workflow management systems as well as general business process management tools that can be used to assemble a workflow. Some common toolkit as well as languages will be discussed:

Kepler [121] is a popular free scientific workflow management system for designing, executing, reusing, evolving, archiving, and sharing workflows.Kepler offers the following functionalities including: process & data monitoring, provenance information and high speed data movement solution.Kepler models workflow using directed graphs. A node (Actors) represents a discrete computational component while a edge (channel) represent paths which data or result flows between the actors. The Kepler workflow system comes with a GUI, a runtime engine to execute the workflow from the GUI, commandline or as a distributed computing option.

Taverna Workbench [122] is an open source workflow tool that can design as well as execute workflows. Taverna was created by the myGrid project and funded through the OMII-UK.Taverna allows users to integrate different software libraries as well as using components with SOAP or REST web services. Taverna comes with an authoring environment as well as a scientific workflow engine which can be downloaded separately as a standalone server.

visTrails [123] is free open source scientific workflow and provenance management system designed to provide data exploration and visualisation. visTrails offers users to generate and evaluate their data using a series of related but different workflows that can be adjusted

interactively. The aim of visTrails is to manage rapidly evolving workflows that is suited to tasks such as simulations, data analysis and visualisation.

Business Process Execution Language (BPEL) [124] is a OASIS standard executable XML language which specifies action within business process with web services. BPEL uses web services exclusively to transfer information between different processes. BPEL interacts with external entities through web services operations defined using WSDL 1.1.It provides both hierarchical and graph-like control mechanism as well as data manipulation functions. BPEL also supports long running transaction model that can recover failed business processes. There exists several implementation that utilises the BPEL language including: ActiveVOS [125], Open Source Easy BPEL [126], Eclipse BPEL project [127], The Open Source BPMS [128], Apache ODE and BPEL for Windows Workflow Foundation [129].

9.7 Server and Cloud standards

While many existing (e.g. security, virtualization) and emerging standards are important in cloud computing and several emerging efforts towards standardization exist, the adoption of standards in cloud computing is currently low. Some of these, such as security-related standards, apply generally to distributed computing environments. In this section we introduce commonly used definitions and classifications and present several relevant standardization initiatives. In **Recommendations of the National Institute of Standards and Technology** [130] cloud computing is defined as *a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.* The report states that this cloud model promotes availability. Five essential characteristics, three service models, and four deployment models are proposed. All current cloud implementations can be described according to these dimensions.

The essential characteristics are: *On-demand self-service.* A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service's provider. *Broad network access.* Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, and PDAs). *Resource pooling.* The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory, network bandwidth, and virtual machines. *Rapid elasticity.* Capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out, and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time. *Measured Service.* Cloud systems automatically control and optimize resource use by leveraging a metering capability1 at some

level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

The service models described are: *Cloud Software as a Service (SaaS).* The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through a thin client interface such as a web browser (e.g., web-based email). The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings. *Cloud Platform as a Service (PaaS).* The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly application hosting environment configurations. *Cloud Infrastructure as a Service (IaaS).* The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, deployed applications, and possibly limited control of select networking components (e.g., host firewalls).

Finally, the following Deployment Models are defined: *Private cloud.* The cloud infrastructure is operated solely for an organization. It may be managed by the organization or a third party and may exist on premise or off premise. *Community cloud.* The cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be managed by the organizations or a third party and may exist on premise or off premise. *Public cloud.* The cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud services. *Hybrid cloud.* The cloud infrastructure is a composition of two or more clouds (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).

The Open Cloud standards initiative [131], emerging from the Open Grid Forum is among the first to attempt standardization in cloud computing with the Open Cloud Computing Interface specification. Their goal is to develop an open specification and API for cloud offerings. The standards will adhere to the Open Cloud Principles, which are open formats, open interfaces, open data, and open source. . The current focus is on Infrastructure-as-a-Service but the interface can be extended to support Platform and Software as a Service as well.

The Open Cloud Standards Incubator was formed by the the DTMF (Distributed Management Task Force) to assess the impacts of cloud computing on management and virtualization standards and to make recommendations for extensions to better align with the requirements of cloud environments. It aims to enable portability and interoperability between private clouds

within enterprises and hosted or public cloud service providers. The OCSI has published a white paper entitled *Inter-operable Clouds* [132] in which they defined a set of architectural semantics to unify the inter-operable management of enterprise and cloud computing. The report proposes usage scenarios, a service life-cycle and a conceptual Cloud Service Reference Architecture which describes key components (actors, interfaces, data artifacts, and profiles) and the relationships among these components.

The Cloud Management Working Group (CMWG) develops prescriptive specifications that deliver architectural semantics as well as implementation details to achieve inter-operable management of clouds between service requesters/developers and providers. This WG will propose a resource model that at minimum captures the key artifacts identified in the Use Cases and Interactions for Managing Clouds [133] document produced by the Open Cloud Standards Incubator.

The Storage Networking Industry Association (SNIA) proposes the Cloud Data Management Interface (CDMI) [134], which defines the functional interface that applications may use to create, retrieve, update and delete data elements from the Cloud. As part of this interface the client will be able to discover the capabilities of the cloud storage offering and use this interface to manage containers and the data that is placed in them. In addition, metadata can be set on containers and their contained data elements through this interface.

In general, the adoption of cloud and use of cloud resources in different environments may demand compliance with specific requirements of those environments. This would also be the case in healthcare, where due to specific needs concerning security, privacy, legal, availability, performance and conformance with the clinical workflow not all models are applicable and compliance with additional requirements may be necessary. Although not directly cloud-related, the generic requirements for infrastructures for the federal government in the US [135] are a very relevant example to mention here. Their compliance with these additional requirements enabled Google to provide cloud services to the US federal government.

10 Standards relating to specific work packages

10.1 Data warehousing standards

Relevant to WP7

A data warehouse is a multi-component application and service which includes web services, servers, databases & data storage mechanisms, data and data management methods. It is primarily intended to bring together data from many sources in a standardised form to enable advanced analysis (in a business context often described as “business intelligence”). Many methods and standards - for example, web services - have already been mentioned in previous sections and as such, the data warehousing section will focus on data access, storage, provenance and server and cloud standards. The choice of standards relevant to the data warehouse is heavily influenced by the preliminary design for the p-medicine data warehouse. It will be based

around a triplestore, a flexible graph-based data storage mechanism which is an alternative to traditional relational databases queried by SQL. This design decision was motivated by the requirements of semantic standardisation, the Obtima CTMS, and data mining.

10.1.1 Metadata and data

There is general confusion and disagreement about the difference between data and metadata. At the p-medicine Standards Workshop in July 2011, it was generally agreed that the use of the term *metadata* is discouraged, to avoid giving the false impression that it is definitely distinct from data in a well defined way.

10.1.2 Data access and querying

The triplestore concept grew from the requirements for storing **RDF (Resource Description Framework)** data, which is described in detail in Section 9.5.1.

The Object Management Group have specified the **CWM (Common Warehouse Metamodel)**: *"interfaces that can be used to enable easy interchange of warehouse and business intelligence metadata between warehouse tools, warehouse platforms and warehouse metadata repositories in distributed heterogeneous environments"*. It is based upon three other OMG standards:

- UML (Unified Modelling Language)
- MOF (Meta Object Facility)
- XMI (XML Metadata Interchange)

It is supported by many big names in data management, including Oracle, IBM and Unisys, and so supporting it may help our system interoperate with others. However, it is unlikely that such interoperation will be an important use-case, there is debate as to how compliant the corporate supporters are to the standard, there is a debate as to what extent compliance to the standard enables interoperability, and finally the standard is very large (the document is nearly 600 pages long). **XMI**, which is used by the CWM, is an XML-based standard for exchanging metadata, primarily aimed at data about UML models. It is not widely used outside of this application.

SPARQL is a W3C standard for querying RDF data, described in more detail in Section 9.5.5.

10.1.3 Data storage

There are no known standards which apply specifically to this level of data warehouse implementation. There are many opinions on best practice and so on, but little of it is rigorously backed by evidence, and anyway such evidence would be difficult to apply outside of business, the traditional domain of data warehousing.

10.1.4 Standards enabling auditing and curation

A key requirement for the data collected into the warehouse is that it must be auditable and curatable. Annotation, logging and provenance information are required to perform these functions, and many standards exist for storing such information. Since the data in the warehouse will be stored in a triplestore with nouns and verbs taken from a standardised vocabulary or ontology, an obvious option is to store auditing and curation data alongside the collected data as part of the triplestore, such standards feature heavily in this section.

Annotation standards The **DCMI (Dublin Core Metadata Initiatiative)** defines a terminology for basic data which can describe almost any resource, and therefore warrant standardisation. Of over 50 terms in the original **DCMI Metadata Terms**, 15 have been selected to form the **Dublin Core Metadata Element Set** which has become **ISO Standard 15836:2009**. These terms include very basic information, such as creation date, title and subject, resource type and format, and contributing and creating entities. The full set of DCMI terms are less universally applicable, but many could apply to data in the data warehouse. The DCMI is a widely adopted standard and is actively being developed and promoted.

Auditing and curation can be regarded as social processes, and therefore standards which related to social networking are perhaps relevant to these tasks. **FOAF (Friend of a Friend)** describes a set of terms for information about a person which might be used in a social networking context, including full name, nickname, an avatar or other representative image, and relationships between people. An extended terminology defines term for workplace, projects, interests, publications and so on. FOAF is not widely supported by social networks, although Twitter and LiveJournal do publish FOAF information. Interoperability with them is not a motivator for using FOAF. FOAF uses **WebID** URLs to uniquely identify an individual person. **SIOC (Semantically Interlinked Online Communities)** provides an ontology for social networking information, which can be used in conjunction with the FOAF terminology. The main terms which are contributed by SIOC relate to forums, discussion and comments. FOAF, SIOC and WebID are not widely used, therefore there is a question of how mature and stable they are. The primary motivation for using them is to take advantage of the work others have done formulating the terms, although this may be of limited use to p-medicine, because we will only use a subset of the terms.

myExperiment ontology brings together many terminologies and ontologies for describing workflows and experiment plans, the primary use case being the sharing and searching of such workflows in a manner inspired by social networking. Whilst some elements of the ontology are specific to workflows, by drawing an analogy between workflows and datasets, the design and philosophy of the myExperiment sharing website are particularly relevant to the p-medicine data warehouse, and therefore the technologies that underly it are potentially useful. Adopting the myExperiment ontology, and perhaps contributing to it to fulful our specific needs, might allow the myExperiment initiative to expand to cover publication of standardised datasets.

Logging standards Two logging scenarios can help to inform logging in the data warehouse - computer systems event logging, and security audit trails.

RFC 5424 defines the syslog protocol, a layered standard for structuring and transmitting event notifications. Whilst its basic purpose is system logs, it is an extensible protocol. Tools for distributing, analysing and visualising syslogs are available, which would motivate support for this protocol. The protocol is quite simple and does not make use of any of the modern generic data structure standards such as XML or RDF. **CEE (Common Event Expression)** is an attempt at a more expansive standard for logging "computer events". It has yet to define any standards, and is currently at discussion stage, primarily via a mailing list. p-medicine might benefit from being involved in defining this standard at an early stage. **Loggly** is an cloud-based logging service, which provides a complete logging and analysis solution. Use of such a third-party service has some advantages, however it is likely that privacy concerns would override them.

IHE (Integrated Healthcare Initiative) is an organisation which aims to encourage integration of healthcare systems, primarily by encouraging the use and development of existing standards, such as DICOM, HL and W3C. As part of its technical framework, IHE defines **Audit Trail and Node Identification** standards, guidelines and recommendations, which are primarily intended to store auditing information relating to auditing security and privacy. In particular, it defines auditing information to be stored for transactions on DICOM servers. These appear to be useful suggestions, however the detail of the IHE technical framework is vast, impacts upon many areas of p-medicine (and therefore would require more general adoption to be useful within the data warehouse), and would be difficult to follow in p-medicine generally.

Supplement 95 of the DICOM standards defines audit trail messages for DICOM transactions. Since our intention in the data warehouse is to use a pre-existing DICOM server, support for these audit messages will be implementation dependent. Additionally, the usefulness of a domain-specific logging standard is questionable, especially since it creates an undesirable heterogeneous environment for audit and logging analysis tools.

Going some way to solving this problem, IHE recommends the use of **RFC3881**, which defines an XML format for storing information about access to healthcare data, for the purpose of security auditing. This standard primarily relates to access to and updating of active healthcare records - that is, those that are used in hospitals to assist in the treatment of patients. The p-medicine data warehouse is not intended for this purpose, however similar scenarios will exist and support for this standard will allow the use of pre-existing security auditing tools.

Provenance Provenance is closely related to logging and audit trails. The focus in provenance is upon where data has come from originally, rather than what has happened to it since, although there is significant overlap.

The **Open Provenance Model** is a standard for exchanging provenance information about any entity, with the following goals:

- To allow provenance information to be exchanged between systems, by means of a

compatibility layer based on a shared provenance model

- To allow developers to build and share tools that operate on such a provenance model
- To define provenance in a precise, technology-agnostic manner
- To support a digital representation of provenance for any 'thing', whether produced by computer systems or not
- To allow multiple levels of description to coexist
- To define a core set of rules that identify the valid inferences that can be made on provenance representation

The model is based around three types of entity: *agents* which perform *processes* upon *artifacts*. The model attempts to capture the causal relationships between these entities. The model is therefore quite generic, and care must be taken for instances of models expressed within it to be useful and understandable and appropriate to the requirements of p-medicine. The general nature of OPM means it might also be used to store logging information. The OPM ontology is expressed in OWL, and XML format is defined for it, an offical Java toolkit is available, and a small number of tools support OPM.

The **DCMI Metadata Provenance Task Group** is ongoing and has yet to produce any standards. As with the CMM, it may be good for p-medicine to be involved with this standard from an early stage to ensure it is relevant to our requirements.

Provenance of data from scientific experiments and modelling is a growing concern, primarily to aid for repeatability and comparability of experiments, and many projects have built up to standardise it. These are relevant to p-medicine in two ways. Firstly, we may handle data from experiments for which provenance data is supplied in a standard format. Second, we might apply the standards for provenance of experimental data to clinical and other data which p-medicine is to handle. Numerous "Minimum Information" projects have been built up around particular kinds of experimental data, for example MIAME (Minimum Information About a Microarray Experiment - which is described in detail in Section 6.1.6) and MIASE (Minimum Information About a Simulation Experiment - which is described in detail in Section 6.2.1), and **MIBBI (Minimum Information for Biological and Biomedical Investigations)** is a portal for cataloguing these. The important thing to note is that few of the MI projects describe formats or terminologies to standardise the information in machine-readable form - a notable exception being SED-ML which encodes information about a MIASE experiment, described further in Section 6.2.2. Most are plain English descriptions of what should be provided by a human author to make sure experiments are written up in a consistent manner. Therefore, such standards are only partially useful to p-medicine and data warehousing: significant work needs to be done to conform this information into the warehouse for data mining and other analysis tasks.

The **CCLRC Scientific Metadata Model** attempted to define generic provenance and other data about any kind of scientific study and data produced from it. Unlike the Minimum Information projects, this is designed to be machine-readable and defines a UML model and associated XML

schema. This standard was created by the Council for the Central Laboratory of the Research Councils (CCLRC), which has now been merged with the Particle Physics and Astronomy Research Council to create the Science and Technology Facilities Council (STFC). STFC has maintained this standard as the **Core Scientific Metadata Model (CSMD)** which is primarily aimed at defining data about experimental results from the "large facilities". The metadata specified is fairly simple, but it might be useful to follow this standard if we wish to integrate with the ICAT eScience infrastructure in the UK.

10.2 Identity standards

Relevant to WP5

The identity standards will be discussed in more detail in WP5. These standards include specifications for de-identification and entity identification. The most important specifications are listed below.

- ISO TS25237 Health Informatics - Pseudonymization
- Healthcare Information Technology Standards Panel (HITSP)
- Healthcare Services Specification Project
- Entity Identification Service (EIS)
- IHE ITI TF-1 Integration Profiles

10.3 Biobanking standards

Relevant to WP10

Biobanks, sometimes called biorepositories or biomaterial banks represent key resources for clinico-genomic research and advances in personalized medicine. Through the increasing use of molecular and genetic factors in the study of disease causes and targeted individualised therapies biobanks have gained growing importance as part of the scientific infrastructure. The discovery of critical genes and pathways as well as the analysis of their impact and significance will critically depend on sufficient access to biomaterial and related information. In consequence, there is a growing interest in integrating biomaterial repositories into larger infrastructures in order to satisfy research communities' need to specific, quality-assessed samples and up-to-date sample related data with integrated (or at least the possibility to link to) clinical and epidemiological information. A prominent initiative in this direction represents the European Research Infrastructure project BBMRI (Biobanking and Biomolecular Resources Research Infrastructure) whose preparatory phase came to its end in January 2011. Biobank integration into a larger infrastructure or federating biobanks for facilitated access to biomaterial and data requires standardization of various aspects on several levels. Some of them are:

- Biomaterial:
  - Sample quality: Best practices for sample collection, annotation, processing, storage and distribution have to be implemented in order to ensure usability of the biomaterial for research
  - Unique sample identification and cross-laboratory sample tracking mechanisms
  - bar-code label usage standards
- Data
  - Data quality
  - Data harmonization; Terminology and taxonomy for sample classification
  - Data exchange between labs and related data set and communication standards
  - Data linkage with other data sources
  - Linkage of samples and their data to associated research and their results
- Ethical and legal aspects:
  - Privacy protection
  - Access management
  - Role Based Access Control
  - Informed consent and possibilities for later withdrawal or time constraints

While various institutions have issued best-practices-guidelines for the management of biomaterial repositories, which cover only some of the above aspects on the biomaterial level and on the ethical and legal level, these standardisation aspects have not yet been sufficiently addressed for federated biobanks and respective meta biobank infrastructures. BBMRI developed a strategy for communication between biobanks, including a common nomenclature, compatible software techniques and appropriate information transmission policies. The strategy proposes a generalized metadata model for regional metadatabases named BBMRI Hubs. The model holds information about the relation between participating biobanks, their content, user roles and rights. A content meta structure holds a sufficient set of needed information structures and schema elements to boost query answering. Schema elements from local biobanks shall be mapped with the BBMRI Content-Meta structure. The resulting implementation model is a federated hubs and spokes network. A minimum data sat was proposed as an intermediate between BBMRI questionnaires and the generalized metadata model. The dataset is composed of a limited number of attributes describing i) biobanks, ii) studies and iii) subjects, cases and samples within biobanks. The state-of-the-art in federated biobanking is further investigated in WP10 and will be described in deliverable D.10.1.

10.4 Data mining standards

Relevant to WP11

Generally speaking, data mining lacks specific standards with only a few exceptions. It does not rely on proprietary formats but uses standards developed for other purposes.

A review from the Indian Institute of Technology Kanpur [136] provides a comprehensive survey and classification of relevant standards supporting data mining. The data mining standards are classified based on one or more of the following issues:

1. The overall process by which data mining models are produced, used, and deployed: This includes, for example, a description of the business interpretation of the output of a classification tree.
2. A standard representation for data mining and statistical models: This includes, for example, the parameters defining a classification tree.
3. A standard representation for cleaning, transforming, and aggregating attributes to provide the inputs for data mining models: This includes, for example, the parameters defining how zip codes are mapped to three digit codes prior to their use as a categorical variable in a classification tree.
4. A standard representation for specifying the settings required to build models and to use the outputs of models in other systems: This includes, for example, specifying the name of the training set used to build a classification tree.
5. Interfaces and Application Programming Interfaces (APIs) to other languages and systems: There are standard data mining APIs for Java and SQL. This includes, for example, a description of the API so that a classification tree can be built on data in a SQL database.
6. Standards for viewing, analyzing, and mining remote and distributed data: This includes, for example, standards for the format of the data and metadata so that a classification tree can be built on distributed web-based data.

Specific standards are evolving for the first two of these categories.

The 1999 European Cross Industry Standard Process for Data Mining [137] (CRISP-DM 1.0) is an effort to capture the various steps in a data mining process including Business Understanding, Data Understanding, Data Preparation, Modelling, Evaluation, and Deployment.

The Java Data Mining (JDM) is a standard Java API for developing data mining applications and tools. The JDM 1.0 standard was developed under the Java Community Process as JSR 73. As of 2006, the JDM 2.0 specification is being developed under JSR 247 [138]. Various data mining functions and techniques like statistical classification and association, regression analysis, data clustering, and attribute importance are covered by the 1.0 release of this standard.

These are evolving standards; later versions of these standards are under development. Independent of these standardization efforts, freely available open-source software systems like the R language, Weka, KNIME, RapidMiner, jHepWork and others have become an informal standard for defining data-mining processes. Notably, all these systems are able to import and export models in Predictive Model Markup Language.

The Predictive Model Markup Language [139] (PMML) is a more formal standardisation activity concerned with representing data mining models in terms of an XML-based language. It is a vendor-independent method of defining models so that proprietary issues and incompatibilities are no longer a barrier to the exchange of models between applications. It was developed by the Data Mining Group (DMG), an independent group composed of many data mining companies. PMML version 4.0 was released in June 2009.

A PMML model captures several aspects:

- Header
  - Version and timestamp
  - Model development environment information
- Data dictionary defining:
  - Variable types
  - Valid, invalid and missing values
- Data transformations
  - Normalisation, mapping and discretisation
  - Data aggregation and function calls
- Model
  - Description and model-specific attributes
- Mining schema
  - Usage type
  - Outlier and missing value treatment and replacement
- Targets
  - Score post-processing, scaling
  - Definition of model architecture and parameters

Many data mining systems support exchanging PMML models (for a list see Wikipedia [140]). One notes that the support in terms of versions is quite heterogeneous) but the extent of usage of PMML as an exchange mechanisms is not very well documented.

10.4.1 Data Mining Tools and Systems

Notably, data mining rather relies on quasi-standards as defined by data tools and data mining systems rather than on standards. All these systems have internal proprietary data formats for the exchange between the components of the system and for storage. Examples for data mining tools and systems are described below.

RapidMiner[141] RapidMiner provides many data mining and machine learning procedures including data pre-processing and visualization. These can be nested to construct complex data mining processes. It integrates learning schemes and attribute evaluators of the machine learning environment WEKA and statistical modelling schemes of the R-Project.

Weka[142] Weka is an alternative to RapidMiner also providing many machine learning and data mining algorithms.

KNIME[143] KNIME is an open source data analytics, reporting and integration platform that integrates various components for machine learning and data. It provides a graphical user interface allowing quick and easy assembly of nodes for data preprocessing (Extraction, Transformation, Loading), for modelling and data analysis, and visualization.

jHEpWork[144] jHepWork is a full-featured multiplatform data-analysis framework that incorporates many open-source math software packages into a coherent interface using the concept of Java scripting, rather than only-GUI or macro-based concept. jHepWork uses Jython, the Python language for the Java platform in order to call Java numerical and visualization libraries, which brings more power and simplicity for scientific computing. Other scripting languages (like BeanShell etc.) and, of course, Java itself, can also be used.

R[145] R is a programming language and development environment for statistical computing and as a de facto standard for developing statistical software. In Dicode use case 1, R is used extensively but on the level of scripts (see Section 3 of deliverable D2.2). Dicode will split these scripts into reusable services using Rserve as enactment machine. Rserve [146] is a TCP/IP server that allows other programs to use the functionality of R. This allows combining R-based algorithms with other services.

Java Data Mining Package[147] The Java Data Mining Package (JDMP) is an open source Java library for data analysis and machine learning. It facilitates the access to data sources and machine learning algorithms (e.g. clustering, regression, classification, graphical models, optimization) and provides visualization modules. Import and export interfaces are provided for JDBC data bases, TXT, CSV, Excel, Matlab, Latex, MTX, HTML, WAV, BMP and other file formats.

JDMP provides a number of algorithms and tools, but also interfaces to other machine learning and data mining packages.

Other Many other data mining packages are available such as, e.g., Rattle [148], Apache Lucene [149], LibSVM [150] and tm [151] for Text Mining.

10.4.2 Data Exchange Formats

Data mining algorithms typically access data stored in databases using SQL-like languages. Explicit data exchange on file level takes place using proprietary formats like that of Excel or the ubiquitous comma separated values [152] (CSV), typically with a first row consisting of attribute names and subsequent rows being corresponding values. Another common file format is ARFF [153] (Attribute-Relation File Format). ARFF file is an ASCII text file that describes a list of instances sharing a set of attributes and was initially developed for the use with Weka but currently is supported by other data mining tools such as Rattle and RapidMiner.

10.4.3 Data Mining Performance

Performance measurement of data mining relies on benchmark data sets as, for instance, provided by the UC Irvine Machine Learning Repository [154] or the Edinburgh Data Sets for Data Mining [155]. Alternatively, many data mining conferences or institutions set up data mining challenges or contests (see, e.g., ECML PKDD 20 Discovery Challenge [156], Hearst Challenge 2011 [157]).

10.4.4 Large Scale Data Mining

Data mining on really large data sets is demanding in terms of computation time and storage that often exceeds the capacity of a single work station (with, e.g., a storage of even 128 Gbyte and computation time of, e.g., 4 month). Distributed data mining is a possible option for speed up and more storage. Several systems are under development with Mahout possibly developing into a quasi-standard though there are competing systems.

Mahout[158] Apache Mahout currently has a very active development community. It has been reported to be used as part of the spam filtering pipeline at Yahoo! [159], for matching couples at Speeddate, as well as a part of the recommendation modules at AOL and Foursquare. Mahout's goal is to provide scalable machine learning libraries for the Java programming language. Mahout implements most of its algorithms on top of Apache Hadoop [160] using the Map/Reduce paradigm. Therefore, it is perfectly suited for massive data that is stored in a Hadoop Cluster.

A very recent project called Radoop [161] integrates Hadoop in RapidMiner by providing a Hadoop extension for RapidMiner. Since Mahout uses Hadoop, it will then be possible to use

Mahout class library from within RapidMiner.

Twister[162] Twister is as well a Map/Reduce implementation. It uses pub/sub messaging for all the communication/data, eliminating the need for a specialised distributed file system. If the computational resources required by an algorithm are large in relation to the data volume, a considerable speed-up can be observed in comparison to Hadoop.

R Several packages support distributed computing in R. For instance, the R-Package GridR [163] allows submitting R functions for execution on remote computers, clusters, or grids.

A List of Requirements Management (RM) tools

- RaQuest [164]
- Cradle [165]
- CORE [166]
- CMS & RTS [167]
- IBM Rational Software [168]
- Borland CaliberRM [169]
- Acclaro DFSS [170]
- Accept 360 [171]
- Blueprint Requirements Center 2010 [172]
- Cognition Cockpit [173]
- GatherSpace [174]
- IRqA [175]
- Leap SE [176]
- PACE [177]
- Polarion REQUIREMENTS [178]
- SpiraPlan [179]
- TestTrack RM [180]
- Jama Contour [181]

- Enterprise Architect [182]
- Lighthouse RM [183]
- MKS Requirements [184]
- Open Source RM [185]
- Projectricity [186]
- SoftREQ [187]
- TopTeam Analyst [188]
- Accompa [189]
- CASE Spec [190]
- objectiF [191]
- RALLY Agile [192]
- RMTrak [193]
- TRUEreq [194]
- ARCWAY Cockpit [195]
- Bright Green Projects [196]
- Case Complete [197]
- OneDesk requirements management [198]
- Modelio Requirement Analyst [199]
- Objectiver [200]

B List of Open Source licenses

License	Author
Academic Free License	Lawrence E. Rosen
Affero GPL	Free Software Foundation
Apache License	Apache Software Foundation
Apple Public Source License	Apple Computer
Artistic License	Larry Wall
Berkeley Database License	Oracle Corporation
BSD license	Regents of the University of California
Boost Software License	?
Common Development and Distribution License	Sun Microsystems
Code Project Open License	The Code Project
Common Public License	IBM
Cryptix General License	Cryptix Foundation
Eclipse Public License	Eclipse Foundation
Educational Community License	?
Eiffel Forum License	NICE
EUPL	European Commission
Fair Licence	?
GNU General Public License	Free Software Foundation
GNU Lesser General Public License	Free Software Foundation
Hacktivismo Enhanced-Source Software License Agreement	Hacktivismo/Cult of the Dead Cow
IBM Public License	IBM
Intel Open Source License	Intel Corporation
ISC license	Internet Systems Consortium
LaTeX Project Public License	LaTeX project
MIT license / X11 license	MIT
Mozilla Public License	Mozilla Foundation
Netscape Public License	Netscape
OPaC Free Public License	OPaC bright ideas
Open Software License	Lawrence Rosen
OpenSSL license	OpenSSL Project
PHP License	PHP Group
Python Software Foundation License	Python Software Foundation
Q Public License	Trolltech
Sun Industry Standards Source License	Sun Microsystems
Sun Public License	Sun Microsystems
Sybase Open Watcom Public License	?
W3C Software Notice and License	?
XCore Open Source License	XMOS
XFree86 1.1 License	?
zlib/libpng license	?
Zope Public License	?

References

[1] http://toolkit.vph-noe.eu/toolkit-guidelines

[2] http://toolkit.vph-noe.eu/component/docman/doc.download/2-g01-toolkit-tool-characterisation-guideline-v10

[3] http://toolkit.vph-noe.eu/component/docman/doc.download/3-g02-toolkit-model-characterisation-guideline-v10

[4] http://toolkit.vph-noe.eu/component/docman/doc.download/8-g03-toolkit-data-characterisation-guideline-v10

[5] http://toolkit.vph-noe.eu/component/docman/doc.download/4-g04-toolkit-ontological-annotation-guideline-v10

[6] http://www.vph-ricordo.eu

[7] http://toolkit.vph-noe.eu/component/docman/doc.download/5-g05-toolkit-interoperability-guideline-v10

[8] http://toolkit.vph-noe.eu/component/docman/doc.download/6-g06-toolkit-ethico-legal-guideline-v10

[9] http://toolkit.vph-noe.eu/component/docman/doc.download/1-g07-toolkit-licensing-guideline-v10

[10] http://toolkit.vph-noe.eu/component/docman/doc.download/7-g08-toolkit-usability-a-training-guideline-v10

[11] http://www.computer.org/portal/web/swebok

[12] http://jucmnav.softwareengineering.ca/ucm/bin/view/UCM/WebHome

[13] http://www.cs.toronto.edu/km/GRL/

[14] http://www.soapatterns.org/

[15] http://martinfowler.com/books.html\#eaa

[16] http://en.wikipedia.org/wiki/Design.pattern..28computer.science.29

[17] http://sourcemaking.com/design.patterns

[18] http://www.iso.org/iso/iso.catalogue/catalogue.tc/catalogue.detail.htm?csnumber=38854

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

[20] http://www.omg.org/mda/

[21] http://www.iso.org/iso/iso.catalogue/catalogue.tc/catalogue.detail.htm?csnumber=21573

[22] http://en.wikipedia.org/wiki/Formal_methods

[23] http://knol.google.com/k/jurgen-appelo/software-development-methodologies/z7e4mx2g6lir/2

[24] http://windows.microsoft.com/

[25] http://www.microsoft.com/visualstudio/en-us

[26] http://www.microsoft.com/net/

[27] http://www.apple.com/macosx/

[28] http://developer.apple.com/technologies/tools/

[29] http://developer.apple.com/technologies/mac/cocoa.html

[30] http://www.java.com/

[31] http://www.eclipse.org/

[32] http://netbeans.org/

[33] http://en.wikipedia.org/wiki/LAMP..28software.bundle.29

[34] http://www.oracle.com/technetwork/java/codeconv-138413.html

[35] http://www.possibility.com/Cpp/CppCodingStandard.html

[36] http://www.python.org/dev/peps/pep-0008/

[37] http://wiki.services.openoffice.org/wiki/Perl.Coding.Standards

[38] http://pear.php.net/manual/en/standards.php

[39] http://www.w3.org/MarkUp/

[40] http://www.gnu.org/prep/standards/

[41] http://en.wikipedia.org/wiki/Free_software_licence

[42] http://en.wikipedia.org/wiki/Comparison_of_free_software_licenses

[43] http://en.wikipedia.org/wiki/ISO.9000

[44] http://en.wikipedia.org/wiki/Capability.Maturity.Model.Integration

[45] http://en.wikipedia.org/wiki/ISO/IEC.15504

[46] http://tools.ietf.org/html/rfc2616

[47] http://tools.ietf.org/html/rfc2818

[48] http://www.w3.org/TR/soap12-part1/

[49] http://www.ibm.com/developerworks/library/specification/ws-secure/

[50] http://www.ics.uci.edu/.fielding/pubs/dissertation/rest.arch.style.htm

[51] http://www.soundcloud.com

[52] http://www.statowl.com/custom.ria.market.penetration.php

[53] http://www.w3.org/TR/WCAG20/

[54] http://www.w3.org/WAI/

[55] http://www.usability.gov/guidelines/

[56] http://www.webstyleguide.com/

[57] http://www.ncrr.nih.gov/publications/informatics/ehr.pdf

[58] http://wiki.ihe.net/index.php?title=Profiles

[59] http://www.cdisc.org

[60] http://www.sas.com/industry/life-sciences/cdisc/index.html

[61] http://www.mged.org/Workgroups/MIAME/miame.html

[62] http://sagebase.org/commons/background.php

[63] http://www.biomodels.net/miase/

[64] http://precedings.nature.com/documents/5846/version/1/files/npre20115846-1.pdf

[65] http://sed-ml.svn.sourceforge.net/

[66] http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf

[67] http://en.wikipedia.org/wiki/Security.Assertion.Markup.Language

[68] http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf

[69] http://projectliberty.org/

[70] http://kantarainitiative.org/

[71] http://www.oasis-open.org/committees/tc.home.php?wg.abbrev=wss

[72] http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/os/ws-trust-1.4-spec-os.pdf

[73] http://docs.oasis-open.org/ws-sx/ws-secureconversation/v1.4/os/ws-secureconversation-1.4-spec-os.pdf

[74] http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/os/ws-securitypolicy-1.3-spec-os.pdf

[75] http://docs.oasis-open.org/wsfed/federation/v1.2/os/ws-federation-1.2-spec-os.pdf

[76] http://openid.net/specs/openid-authentication-2.0.html

[77] http://openid.net/specs/openid-simple-registration-extension-1.0.html

[78] http://openid.net/specs/openid-attribute-exchange-1.0.html

[79] http://openid.net/specs/openid-provider-authentication-policy-extension-1.0.html

[80] http://tools.ietf.org/html/rfc5280

[81] http://www.ietf.org/rfc/rfc3281.txt

[82] http://en.wikipedia.org/wiki/Certification.path.validation.algorithm

[83] http://www.jasig.org/cas/protocol

[84] http://tools.ietf.org/html/rfc1510

[85] http://tools.ietf.org/html/rfc5849

[86] http://www.oasis-open.org/committees/xacml/

[87] http://www.oasis-open.org/committees/download.php/2713/Brief.Introduction.to.XACML.html

[88] http://www-dse.doc.ic.ac.uk/Research/policies/ponder/PonderSpec.pdf

[89] http://ponder2.net

[90] http://sec.cs.kent.ac.uk/download/ModularPermisConcPracExpRevised.pdf

[91] http://gridge.psnc.pl/

[92] http://www.gridlab.org/

[93] http://research.microsoft.com/pubs/76079/UCAM-CL-TR-648.pdf

[94] http://siswg.net/index.php?option=com.content.task=view.id=38.Itemid=73

[95] http://siswg.net/index.php?option=com.content.task=view.id=36.Itemid=75

[96] http://www.oasis-open.org/committees/kmip/

[97] http://siswg.net/index.php?option=com.content.task=view.id=35.Itemid=76

[98] http://www.mozilla.org/projects/security/pki/nss/ssl/draft302.txt

[99] http://tools.ietf.org/html/rfc5246

[100] http://standards.iso.org/ittf/PubliclyAvailableStandards/s020269.ISO.IEC.7498-1.1994.28E.29.zip

[101] http://tools.ietf.org/html/rfc4301\#page-4

[102] http://standards.iso.org/ittf/PubliclyAvailableStandards/s020269.ISO.IEC.7498-1.1994(E).zip

[103] http://ec.europa.eu/information.society/activities/health/docs/publications/2009/2009semantic-health-report.pdf

[104] http://eurlex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32008H0594:EN:NOTEC

[105] *Classification Algorithms Applied to Narrative Reports.* Adam Wilcox, George Hripcsak. 1999. AMIA.

[106] *Classification of Cancer Stages from Histology Reports.* Iain McCowan, Darren Moore, Mary-Jane Fry. 2006. IEEE.

[107] *Three Approaches to Automatic Assignment of ICD-9-CM Codes to Radiology Reports.* Ira Goldstein, Anna Arzumtsyan,Ozlem Uzuner. s.l. : AMIA, 2007.

[108] *The Role of Domain Knowledge in Automating Medical Text Report Classification.* Adam Wilcox, George Hripcsak. s.l. : JAMIA, 2003, JAMIA.

[109] *Categorization of Sentence Types in Medical Abstracts.* Larry McKnight, Padmini Srinivasan. 2003. AMIA. p. 440.

[110] *Machine learning and Rule-based Approaches to Assertion Classification.* Ozlem Uzuner, Xiaoran Zhang,Tawanda Sibanda. s.l. : JAMIA, 2009.

[111] *The Effect of Feature Representation on MEDLINE Document Classification.* Meliha Yetisgen, Wanda Pratt. 2005. AMIA. pp. 849-853.

[112] *Identification of Findings Suspicious for Breast Cancer Based on Natural Language Processing of Mammogram Reports.* Nilesh L.Jain, Carol Friedman. 1997. AMIA. pp. 829-833.

[113] *Automated Encoding of Clinical Documents Based on Natural Language Processing.* Carol Friedman, Lyudmila Shagina,Yves Lussier,George Hripcsak. 2004, JAMIA, pp. 392-402.

[114] http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3097480/pdf/TOMINFOJ-4-235.pdf

[115] http://www.hl7.org/special/Committees/arden/index.cfm

[116] Manola F, Miller E. *RDF Primer.* W3C Recommendation 10 February 2004. http://www.w3.org/TR/rdf-primer/

[117] Brickley D, Guha R.V. *RDF Vocabulary Description Language 1.0: RDF Schema.* W3C Recommendation 10 February 2004. http://www.w3.org/TR/rdf-schema/

[118] Dean M, Schreiber, G. *OWL Web Ontology Language Reference.* W3C Recommendation 10 February 2004. http://www.w3.org/TR/owl-ref/

[119] Motik B, Patel-Schneider P.F, Parsia B. *OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax.* W3C Recommendation 27 October 2009. http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/

[120] Prud'hommeaux E, Seaborne A. *SPARQL Query Language for RDF.* W3C Recommendation 15 January 2008. http://www.w3.org/TR/rdf-sparql-query/

[121] https://kepler-project.org

[122] http://www.taverna.org.uk/

[123] http://www.vistrails.org

[124] http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf

[125] http://activevos.com/products-activevos.php

[126] http://easybpel.petalslink.org/user-guides.html

[127] http://www.eclipse.org/bpel/

[128] http://www.intalio.com/

[129] http://ode.apache.org/

[130] http://csrc.nist.gov/publications/drafts/800-145/Draft-SP-800-145_cloud-definition.pdf

[131] http://occi-wg.org/

[132] http://dmtf.org/sites/default/files/standards/documents/DSP-IS0101.1.0.0.pdf

[133] http://cloud-standards.org/wiki/index.php?title=Main.Page

[134] http://www.snia.org/cdmi

[135] http://csrc.nist.gov/groups/SMA/fisma/index.html

[136] http://www.datamininggrid.org/wdat/works/att/standard01.content.08439.pdf

[137] http://community.udayton.edu/provost/it/training/documents/SPSS.CRISPWPlr.pdf

[138] http://www.jcp.org/en/jsr/detail?id=73

[139] http://www.dmg.org/v4-0-1/GeneralStructure.html

[140] http://en.wikipedia.org/wiki/Predictive_Model_Markup_Language

[141] http://rapid-i.com/content/view/181/190/lang,en/

[142] http://en.wikipedia.org/wiki/Weka.(machine.learning)

[143] http://www.knime.org/

[144] http://jwork.org/jhepwork/

[145] http://en.wikipedia.org/wiki/R-Project

[146] http://www.rforge.net/Rserve/

[147] http://www.jdmp.org/

[148] http://rattle.togaware.com/

[149] http://lucene.apache.org/java/docs/index.html

[150] http://www.csie.ntu.edu.tw/.cjlin/libsvm/

[151] http://cran.r-project.org/web/packages/tm/index.html

[152] http://en.wikipedia.org/wiki/Comma-separated.values

[153] http://weka.wikispaces.com/ARFF+.28stable+version.29

[154] http://archive.ics.uci.edu/ml/

[155] http://www.inf.ed.ac.uk/teaching/courses/dme/html/datasets0405.html

[156] http://www.ecmlpkdd2011.org/challenge.php

[157] http://hearstchallenge.com/

[158] http://mahout.apache.org/

[159] http://www.slideshare.net/hadoopusergroup/mail-antispam

[160] http://hadoop.apache.org/

[161] http://prekopcsak.hu/index.php?slug=radoop

[162] http://www.iterativemapreduce.org/

[163] http://cran.r-project.org/web/packages/GridR/

[164] http://www.raquest.com/

[165] http://www.threesl.com/pages/Cradle/English/Content/Products/overview.php

[166] http://www.vitechcorp.com/products/index.html

[167] http://pcwin.com/Business...Finance/Applications/Requirements.Tracing.System/index.htm

[168] http://www-01.ibm.com/software/rational/offerings/irm/

[169] http://www.borland.com/us/products/caliber/

[170] http://www.dfss-software.com/

[171] http://www.accept360.com/

[172] http://www.blueprintsys.com/products/

[173] http://www.cognition.us/cockpit.overview.html

[174] http://www.gatherspace.com/

[175] http://www.visuresolutions.com/irqa-requirements-tool

[176] http://www.leapse.com/

[177] http://www.viewset.com/

[178] http://www.polarion.com/products/requirements/index.php

[179] http://www.inflectra.com/SpiraPlan/Default.aspx

[180] http://www.seapine.com/ttrm.html

[181] http://www.jamasoftware.com/contour/

[182] http://www.sparxsystems.com/platforms/requirements.management.html

[183] http://www.workspace.com/workspace/requirements.html

[184] http://www.mks.com/solutions/discipline/rm/requirements-management

[185] http://sourceforge.net/projects/osrmt/

[186] http://www.projectricity.com/requirements.management.tool.htm

[187] http://www.softreq.com/features.cfm

[188] http://www.technosolutions.com/topteam.requirements.management.html

[189] http://www.accompa.com/requirements-management-software.html

[190] http://www.casespec.net/requirementsmanagement.htm

[191] http://www.microtool.de/instep/en/grafisches.anforderungsmanagement.asp

[192] http://www.rallydev.com/agile.products/lifecycle.management/

[193] http://www.rmtrak.com/

[194] http://www.truereq.com/requirement-management.html

[195] http://www.arcway.com/en/product/arcway-cockpit-3/

[196] http://www.brightgreenprojects.com/

[197] http://www.casecomplete.com/

[198] http://www.onedesk.com/features/requirements-management/

[199] http://www.modeliosoft.com/en/modules/modelio-requirement-analyst.html

[200] http://www.objectiver.com/