



Deliverable No. 8.1

Design of the CHIC repositories

Grant Agreement No.: 600841
Deliverable No.: D8.1
Deliverable Name: Design of the CHIC repositories
Contractual Submission Date: 31/07/2014
Actual Submission Date: 21/11/2014

Dissemination Level		
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)	X



COVER AND CONTROL PAGE OF DOCUMENT	
Project Acronym:	CHIC
Project Full Name:	Computational Horizons In Cancer (CHIC): Developing Meta- and Hyper-Multiscale Models and Repositories for <i>In Silico</i> Oncology
Deliverable No.:	D8.1
Document name:	Design of the CHIC repositories
Nature (R, P, D, O) ¹	R
Dissemination Level (PU, PP, RE, CO) ²	CO
Version:	1
Actual Submission Date:	21/11/2014
Editor: Institution: E-Mail:	Fay Misichoni ICCS-NTUA faymisi@mail.ntua.gr

ABSTRACT:

The present document describes the design of the CHIC repositories. More specifically, this report presents the design of the following repositories:

- The clinical data repository, which will permanently host all the related medical data produced or collected by the CHIC project.
- The model/tool repository, which will host the hypomodels, the hypermodels, the data transformation tools, the linkers and other tools used in the construction and execution of hypermodels.
- The *in silico* trial repository, where the data related to *in silico* trials and *in silico* experiments will be stored.
- The RDF storage solution for semantic metadata.

KEYWORD LIST:

repository, clinical data, model, hypermodel, hypomodel, *in silico* trial, semantics, metadata, security, RDF, *in silico* medicine, repeatability, reproducibility

¹ R=Report, P=Prototype, D=Demonstrator, O=Other

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

The research leading to these results has received funding from the European Union Seventh Framework Programme under grant agreement n° 600841.

The author is solely responsible for its content, it does not represent the opinion of the European Community and the Community is not responsible for any use that might be made of data appearing therein.

MODIFICATION CONTROL			
Version	Date	Status	Author
0.1	03/07/2014	Index Draft	Fay Misichroni, ICCS-NTUA
0.2	15/07/2014	Draft	Roman Niklaus, UBERN
0.3	18/07/2014	Draft	Philippe Büchler, UBERN
0.4	25/07/2014	Draft	Bernard de Bono, UCL
0.5	15/08/2014	Draft	Fay Misichroni, ICCS-NTUA
0.6	22/9/2014	Draft	Fay Misichroni, ICCS-NTUA
0.7	27/10/2014	Draft	Fay Misichroni ICCS-NTUA
0.8	19/11/2014	Draft	Fay Misichroni, ICCS-NTUA
0.9	20/11/2014	Revision	Dimitra Dionysiou, ICCS-NTUA
1.0	20/11/2014	Final	Georgios Stamatakos, ICCS-NTUA

List of contributors

—

Contents

CONTENTS.....	5
FIGURES.....	6
TABLES.....	6
1 EXECUTIVE SUMMARY.....	7
2 INTRODUCTION	8
2.1 PURPOSE OF THIS DOCUMENT	8
2.2 STRUCTURE OF THE DELIVERABLE.....	8
3 CLINICAL DATA REPOSITORY	10
3.1 INTRODUCTION	10
3.2 DATA FLOW AND INTERACTION	10
3.3 DATA	11
3.3.1 <i>Medical imaging data</i>	11
3.3.2 <i>Clinical study data</i>	13
3.3.3 <i>Histological data</i>	14
3.3.4 <i>Genomics data</i>	14
3.4 GENERAL CONCEPTS	15
3.4.1 <i>Linking</i>	15
3.4.2 <i>Annotation & Search</i>	16
3.4.3 <i>Validation & Versioning</i>	17
3.4.4 <i>Data organization</i>	17
3.4.5 <i>Data storage</i>	18
3.5 SOFTWARE ARCHITECTURE AND DESIGN.....	19
3.5.1 <i>Presentation layer</i>	19
3.5.2 <i>Business layer</i>	21
3.5.3 <i>Data access layer</i>	21
3.5.4 <i>Persistence layer</i>	22
3.5.5 <i>Security layer</i>	23
3.6 DOMAIN MODEL.....	23
3.7 TECHNOLOGY.....	26
3.8 SECURITY	27
3.8.1 <i>Role-based security</i>	28
3.8.2 <i>Object-based security</i>	28
3.9 DATA REPRESENTATION	30
4 MODEL/TOOL REPOSITORY AND <i>IN SILICO</i> TRIAL REPOSITORY	33
4.1 INTRODUCTION	33
4.2 MODEL/TOOL REPOSITORY DATA LAYER.....	34
4.3 REPEATABILITY AND REPRODUCIBILITY	41
4.4 <i>IN SILICO</i> TRIAL REPOSITORY DATA LAYER	42
4.5 AUTHENTICATION AND AUTHORIZATION	49
4.6 IMPLEMENTATION PRINCIPLES AND SOFTWARE CHOICES	51
4.7 SOFTWARE ARCHITECTURE	52
4.8 SOFTWARE COMPONENTS	53
4.9 DATA FLOW AND INTEGRATION.....	56
5 RDF STORAGE SOLUTION FOR SEMANTIC METADATA.....	57
5.1 INTRODUCTION	57
5.2 RECORDING FORMAT FOR SEMANTIC METADATA.....	57
5.3 RDF STORE	58
5.4 INTERFACE TO THE STORE	58
5.5 SEMANTIC METADATA IN CONTEXT (FOR INFORMATION ONLY)	60

5.6	SEMANTIC METADATA LIFECYCLE IN CHIC.....	61
6	CONCLUSION	63
7	REFERENCES	65
	Appendix – Abbreviations and acronyms	66

Figures

Figure 1: Data flow and interaction of the clinical data repository.	11
Figure 2: The simplified DICOM data model.	12
Figure 3: DICOM, MetaImage, Analyze and Niftii formats.	13
Figure 4: The CDISC ODM study metadata structure (left) and patient data structure (right).	13
Figure 5: Example of the linking to relate data objects in the clinical data repository for a multi-modal brain segmentation. In this case, four different MRI image datasets are used for the segmentation of brain tumours.	16
Figure 6: Data organization	18
Figure 7: The clinical data repository uses a Data table to store the information about the file, which includes the physical location of the file, its size, name, upload time and hash code. This information is used to ensure the consistency of the data stored in this repository.	19
Figure 8: The multi-tier architecture of the clinical data repository	21
Figure 9: The domain model of the clinical data repository with domain classes (blue), domain enumerations (brown) and their relationships represented as connecting lines. A brief description of each entry is given in the table below.	24
Figure 10: Object-based security in the clinical data repository.	30
Figure 11: Clinical data repository mock-up of the data list view.	31
Figure 12: Clinical data repository mock-up of the data detail view.	32
Figure 13: In silico experiment: Triple of model/hypermodel - input dataset – output dataset	33
Figure 14: In silico trial: Group of in silico experiments in which the same model/hypermodel has been used.	33
Figure 15: The basic entities involved in the model/tool repository.	34
Figure 16: Entity-Relationship (ER) diagram of model/tool repository.	40
Figure 17: Repeatability	41
Figure 18: Reproducibility	42
Figure 19: Entity-Relationship (ER) diagram of in silico trial repository.	48
Figure 20: Entity-Relationship (ER) diagram of tables associated with basic authentication and authorization.	50
Figure 21: Model -View - Controller (MVC) paradigm.	51
Figure 22: Software architecture of model/tool repository and of in silico trial repository.	52
Figure 23: Data flow and interaction between CHIC components.	56
Figure 24: Example of SPARQL query in the RICORDO implementation.	59
Figure 25: Example of REST invocation in the RICORDO implementation.	59
Figure 26: Example of Web-based interface (querying) from RICORDO implementation.	60
Figure 27: Example of desktop interface (annotation) from DDMoRe implementation.	61
Figure 28: Example of Web-based interface (annotation).	61

Tables

Table 1: Domain class description of the clinical data repository.	24
Table 2: Third-party libraries, frameworks and technologies used by the clinical data repository.	27
Table 3: Object-based security domain class description of the clinical data repository.	29

1 Executive Summary

Developing robust, reproducible, interoperable and collaborative hypermodels of diseases and normal physiology is a sine qua non necessity if rational, coherent and comprehensive exploitation of the invaluable information hidden within human multiscale biological data is envisaged. Responding to this imperative in the context of both the broad Virtual Physiological Human (VPH) initiative and the paradigmatic cancer domain, CHIC proposes the development of a suite of tools, services and secure infrastructure that will support accessibility and reusability of VPH mathematical and computational hypermodels. The CHIC tools, services, infrastructure and repositories will provide the community with a collaborative interface for exchanging knowledge and sharing work in an effective and standardized way. A number of open source features and tools will enhance usability and accessibility. In order to ensure clinical relevance and foster clinical acceptance of hypermodelling in the future, the whole endeavour will be driven by the clinical partners of the consortium.

Work package 8 (WP8) focuses on the development of various kinds of repositories, including the design and implementation of the corresponding infrastructures and interfaces.

This involves the development of:

- a repository of cancer models, spanning from models of generic fundamental biomechanisms involved in cancer progression and treatment response, such as cell cycle and cell metabolism, to complex multiscale models of various types of cancer,
- a repository of multiscale data exploitable by the models, either by physically storing the data in the project's data repository, or by providing links to other, already existing, data repositories or warehouses,
- a repository of *in silico* trials for various types of cancer,
- a distributed RDF repository to store metadata from each partner, including the corresponding interfaces for annotating and querying.

The aforementioned repositories will be tailored to the needs/clinical scenarios of the project. At the same time they will be generic enough to be reusable by several different medical scenarios.

2 Introduction

2.1 Purpose of this document

The present document, corresponding to the deliverable “D8.1: Design of the CHIC repositories”, describes the design of:

- the clinical data repository, which will permanently host all the related medical data produced or collected by the CHIC project,
- the model/tool repository, which will host the hypomodels, the hypermodels, the data transformation tools, the linkers and other tools used in the construction and execution of hypermodels,
- the *in silico* trial repository, where the data related to *in silico* trials and *in silico* experiments will be stored,
- the RDF storage solution for semantic metadata.

2.2 Structure of the Deliverable

Chapter 3 focuses on the clinical data repository of CHIC project. The chapter starts with a short description of the data flow and the interactions between the clinical data repository and the other components of the CHIC architecture. The presentation of the different medical data that will be stored in the repository, such as medical imaging data, clinical study data, histopathological data and genomic data follows. The chapter continues with the presentation of the software architecture and the design of the clinical data repository, including the presentation layer, the business layer, the data access layer, the persistence layer and the security layer. General concepts such as versioning, annotations, linking of data are also addressed in this chapter.

Chapter 4 focuses on the model/tool repository and the *in silico* trial repository. This chapter includes a short description of the basic principles of the model/tool repository and the *in silico* trial repository, followed by a detailed presentation of the tables and the fields that are included in each repository. The corresponding Entity-Relationship (ER) diagrams are presented. The chapter

continues with the presentation of the concepts of repeatability and reproducibility which play crucial role in the *in silico* medicine domain. The final part of the chapter focuses on the concepts and the software components used in the implementation of the aforementioned repositories, both the backend and the frontend. The key open source software that has been selected is presented.

In Chapter 5 we indicated the design for the semantic metadata repository as part of the CHIC system. This repository takes the form of an RDF database supporting basic metadata management functionalities. The metadata repository is an element of the metadata CHIC framework, which will include a knowledge base of ontologies supporting reasoning and query expansion of the metadata repository.

3 Clinical Data Repository

3.1 Introduction

The clinical data repository will permanently host all the related medical data produced or collected by the CHIC project. The data will not be directly provided by the clinical environment. The data will pass through de-identification and (pseudo)-anonymization processes, as described in WP4. Additionally, interfaces that will allow to import and export the contents of the clinical data repository will be developed. In this way the data can be sustained after the expiration of the project's lifetime and reused and exploited continuously within the limits allowed by the legal framework of the project. The export services that will be created will also assist in this direction, as many of the data sets to be gathered by the CHIC project will be reusable by future projects. The clinical data repository will contain for each patient all the relevant medical data including imaging data, clinical data, histological data and genetic data.

3.2 Data flow and interaction

The conceptual workflow of the data has been described in deliverable 5.1. In summary, the clinical data are not directly uploaded to the repository by the hospitals, but by an intermediate Trusted Third Party (i.e. CUSTODIX). This intermediate party is responsible to run a pseudonymization service, which will also upload clinical data into the repository. Modellers, researchers, and other experts will have access to read (view) the stored clinical data from the CHIC portal. The hypermodelling infrastructure and the visualization/image processing toolkit will also only have read-only access to the clinical data for their processing purposes. For authentication and authorization the EU-CHIC security framework will be used. Semantic services will also be used by the clinical data repository for annotation and search purposes.

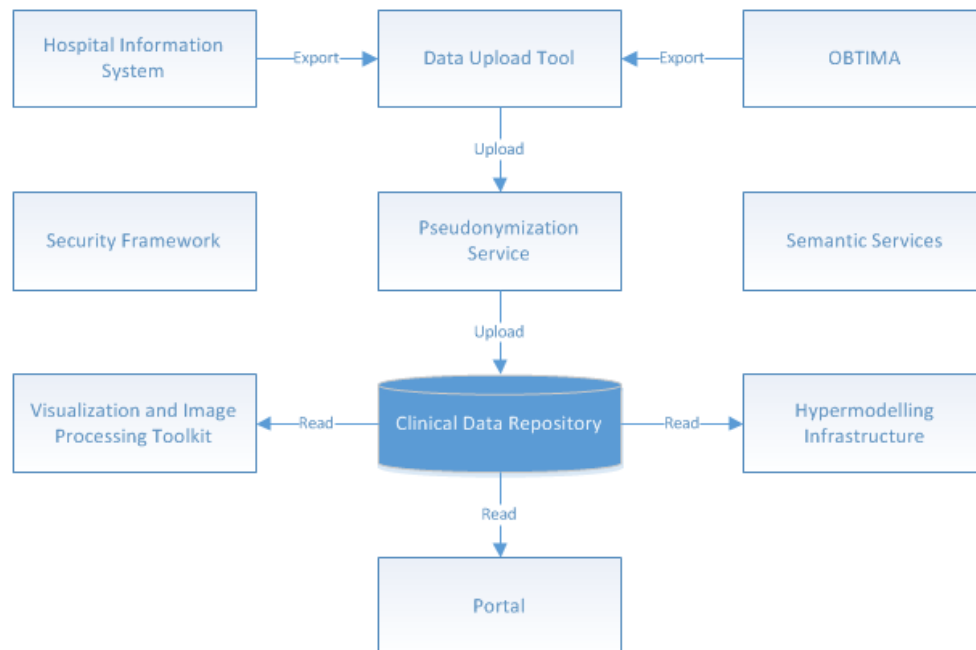


Figure 1: Data flow and interaction of the clinical data repository.

3.3 Data

The concept underlying the design of the clinical data repository is to rely on generic objects. The chosen approach ensures the ability of the system to handle new types of objects with minimal additional effort. However, even though the design is very generic, the structure of the data to be stored is used to provide appropriate verification of the consistence of the data and to extract all the relevant meta-data associated with each uploaded object. In the following sections, a brief description of the major types of objects implemented within the CHIC project is provided.

3.3.1 Medical imaging data

The standard format used in the clinical setting to store the medical images collected by the radiologists, is called “Digital Imaging and Communications in Medicine (DICOM)” [1]. Almost all the medical imaging devices (CT scan, MRI, US, etc.) support that format as well as the PACS systems used by the hospital to manage their clinical data. This standard specifies the methods for storing, printing, and transmitting information in medical imaging. The DICOM data model in its most simplified form is illustrated in Figure 2.

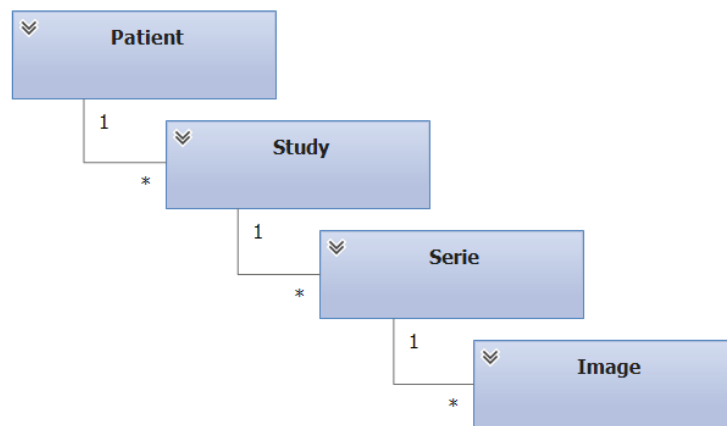


Figure 2: The simplified DICOM data model.

All DICOM objects must include the four main modules called *Patient*, *Study*, *Series* and *Image*. Because every DICOM object must be part of a *Series*, all DICOM objects must include the general *Serie* module and because all *series* must be part of a *Study*, every DICOM object must include the general *Study* module and because every *study* is made on some patient, all DICOM objects must have a *Patient* module. DICOM makes extensive use of unique identifiers. Almost every entity in the DICOM data model has a unique identifier. All unique identifiers, used within the context of the DICOM standard, are registered values as defined by ISO 9834-1 to ensure global uniqueness. The clinical data repository will use these unique identifiers for linking and searching purposes.

The clinical data repository will also be able to store other file formats used by clinicians and researchers. The file formats MetaImage [2], Analyze [3] and Niftii [4] will be supported and will be handled in a similar way as the DICOM file format. The major difference is the amount of files needed to store image information. The DICOM file format allows a single *Serie* to have an arbitrary number of files (usually a few hundreds), with each file containing exactly the image of one slice. The MetaImage file format allows two different approaches. One approach is to store the metadata in a separate header file and all the image information in a separate source file. The second approach is to store the metadata and all images in the same file. The Analyze file format stores the metadata in a separate header file and all image data in a source file. The Niftii file format stores the metadata and all image information as a single file.

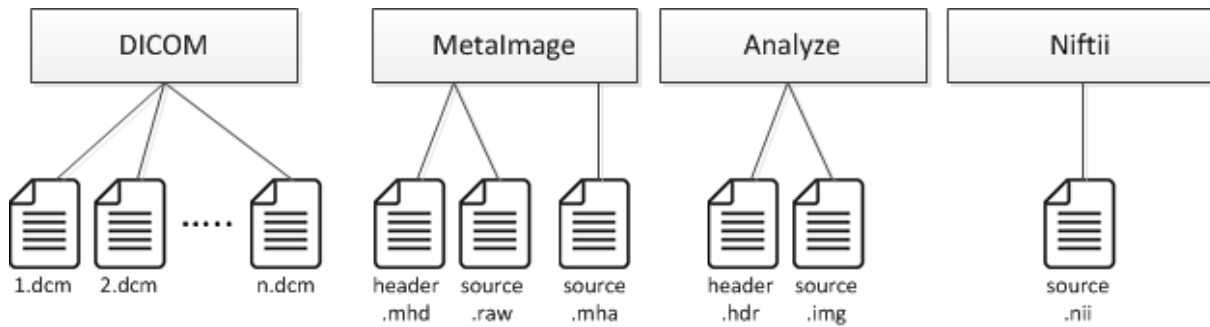


Figure 3: DICOM, Metalmage, Analyze and Niftii formats.

3.3.2 Clinical study data

Besides medical images the clinical data repository will also support clinical study data. For clinical study data, a standard file format is called CDISC (Clinical Data Interchange Standards Consortium) ODM (Operational Data Model) [5] which is designed to facilitate the regulatory-compliant acquisition, archive and interchange of metadata and data for clinical research studies. ODM is a vendor neutral, platform independent format for interchange and archive of clinical study data. The model includes the clinical data (see Figure 4) along with its associated metadata (see Figure 4), administrative data, reference data and audit information. All of the information that needs to be shared among different software systems during the study setup, operation, analysis, submission or for long term retention as part of an archive is included in the model. This makes it the format of choice for the exchange of clinical study data within CHIC.

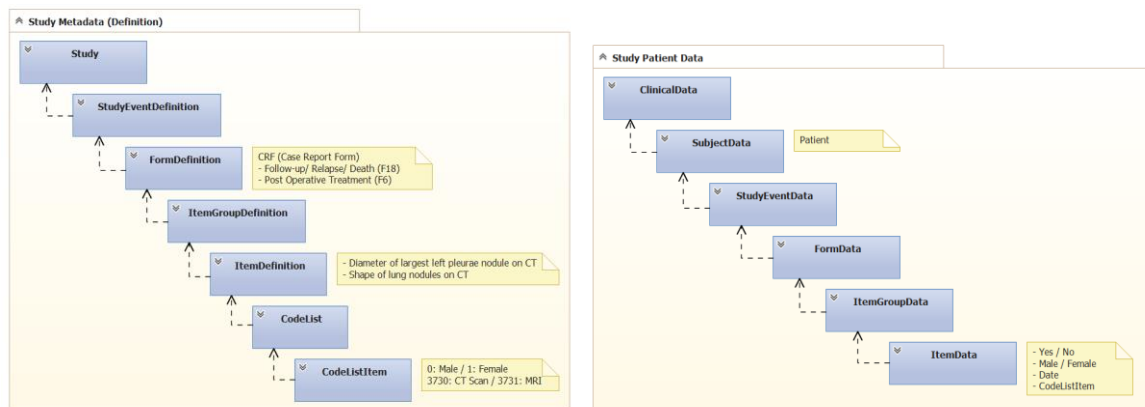


Figure 4: The CDISC ODM study metadata structure (left) and patient data structure (right).

In the ODM model, there are many instances where one element needs to reference another - both within the same file and across files within a series of ODM documents. To accomplish this, the target element is given a specific object identifier (OID) and all elements that need to reference that target element just use its OID. As with the DICOM identifiers, the clinical data repository will use the object identifiers for linking and searching purposes.

3.3.3 Histological data

Whole Slide Imaging

Whole slide imaging is the creation of a single, high magnification digital image of an entire microscopic slide. An automated microscope scans an entire slide at one or more resolutions and digitally concatenates the consecutive small images together into a single large image [6]. No standard file format exists for Whole Slide Imaging (WSI) but four non-proprietary/open formats can be used:

- DICOM standard (supplement 145) approved in 2010; no reference implementation yet
- JP2, the file container for JPEG2000 codestreams
- TIFF/BigTIFF with JPEG compression
- Deep Zoom images

The most frequent file formats will be supported by the clinical data repository to store and exchange the results of histologic examinations.

3.3.4 Genomics data

The Functional Genomics Data Society (FGED) is promoting open access to genomic datasets, which implies the development of standards for biological research annotation and exchange. Their efforts include the standardization of the procedure to report the experimental data, which resulted in the Minimum Information About a Microarray Experiment (MIAME) and the Minimum Information about a high-throughput SEQuencing Experiment (MINSEQE). However, these guidelines do not specify a

specific file format for data exchange. It is clear that the benefit of the data is larger if the information is encoded, using standards, which can be understood by all researchers. MAGE-ML format has been developed to exchange microarray data within the research community. However, due to the complexity of the format, a simplified standard has been developed based on simple tab-delimited, spreadsheet-based format. The MAGE-TAB has been designed to enable laboratories without bioinformatics experience or support to manage, exchange and submit well-annotated microarray data in a standard format using a spreadsheet. For these reason, the clinical data repository will support tabulated text files for the storage and exchange of genomic data.

3.4 General Concepts

The clinical data repository will be built around the concept of data objects (*ObjectVersion*), which constitute the basic component of the system. These data objects can be any type of image file, processed data, study data etc. This approach provides a large flexibility to the system in terms of data formats, data organization and data exchange [7].

The system has been designed to support versioning. Data uploaded to the system are never deleted, but multiple versions of an object can be stored in the database. This approach limits problems associated with accidental deletion of data, while maintaining the flexibility to keep updating data files. For example, the initial data of the clinical study concerning a patient can be uploaded before the final examinations. Once the last examination has been performed, a new version of the file is uploaded to the system, which enables modellers to have access to the latest information while keeping the ability to see the history of the modifications.

3.4.1 Linking

Each new dataset can be linked with any object already present in the repository. For example anatomical structures can be segmented out of one or multiple medical images. Linking mechanisms ensure that an uploaded segmentation file is not only associated with the correct patient's data, but also that the original images used to perform the segmentation task can be identified by the users of the system. In the case of multimodal image segmentation, this implies that multiple links are created to relate the segmentation file with each of the multi-modal original images. If available, the

system will make use of the meta-information stored in the files to automatically generate this linking. The implementation of a manual linking procedure is also planned.

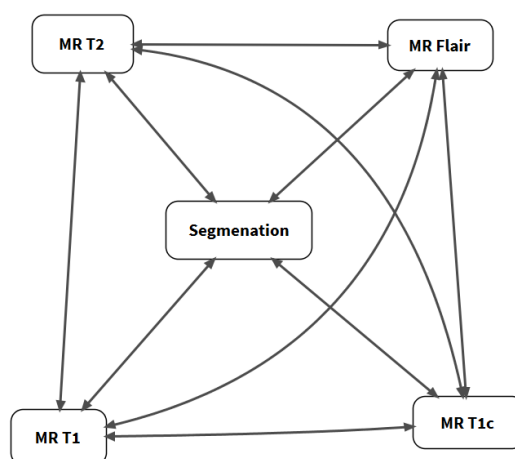


Figure 5: Example of the linking to relate data objects in the clinical data repository for a multi-modal brain segmentation. In this case, four different MRI image datasets are used for the segmentation of brain tumours.

3.4.2 Annotation & Search

In addition to the imaging and clinical data, each data object can be annotated with multiple ontology terms. Initial investigations have been made to integrate an anatomical ontology; the Foundational Model of Anatomy (FMA) [8]. The FMA is a symbolic representation of the canonical, phenotypic structure of an organism; a spatial-structural ontology of anatomical entities and relations which form the physical organization of an organism at all salient levels of granularity. The ontology relies on a triplestore storage system and not in relations or tables. Therefore, a separate system will be used to store the semantic information. Web based queries based on SPARQL [17] will be used to retrieve the information from the ontology for annotation and semantic search. The approach is very flexible and allows to easily include multiple ontologies. In addition to the FMA, additional ontologies will be included in a second step by integrating the RICORDO [13] system. Based on these annotations it will be possible to conduct semantically driven search queries to find datasets containing the required anatomical structures or other properties.

3.4.3 Validation & Versioning

To ensure a high level of quality to the data stored in the repository, the system will support a multi-step validation process. During the validation process the user can review the metadata extracted from the data, include additional relevant information and finally publish the data object. Once published, the new data object is accessible by the other users of the system having the appropriate permissions.

The objects stored in the database cannot be changed or modified once the validation step has been completed. However, the modifications to the objects will be shown as a new version of the data. Versioning makes use of a *GenericObject*; a new version of a dataset will have the same *GenericObject* as the ancestor. An example where a new version of the object will be generated corresponds to the modification in the definition of a clinical study. In case new questions or time points have been added to the study, a new version of the file will be generated. The clinical study will have the same unique identifier as the original object, and based on this identifier a new version can be created. With this approach, the users will still have access to the original data if needed by their models, but the new version will be shown as the current version of the data.

3.4.4 Data organization

The system will also allow the user to freely organize the data that they are using. To this aim, the user interface will allow the creation of virtual folders to organize the datasets. This functionality will provide the user with the flexibility to freely create and organize their personal workspace. Hereby, data objects are not physically moved or duplicated, but the system creates a reference to the data object, retaining the original file permission and ownership. For collaborations or other purposes, the user can share parts of his/her workspace by changing the permission of his/her folder accordingly. To simplify the collaboration within a group, the system provides a default shared group folder, which is accessible and manageable by all members of the group.

As presented in Figure 6, the clinical data repository will allow each user to freely organize the data into his/her desired folder structure for easy access to the data needed for his/her research. The structure created by one user can be directly shared to other co-workers. Modifications made by one

user will immediately be visible in the folder of the other collaborators. The mechanism should allow efficient collaboration between modellers working on the same tumour model.

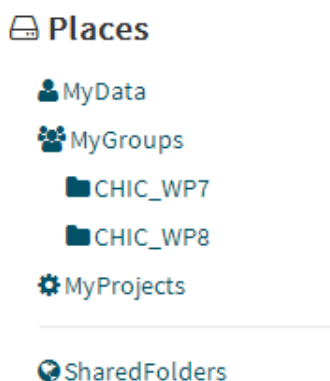


Figure 6: Data organization

3.4.5 Data storage

All files uploaded to the clinical data repository are represented in a *Data table*. The file itself will not be stored in the database but rather on the file system attached to the clinical data repository. The *Data table* contains the information on how the file can be retrieved. This can be best explained with a sample DICOM file as shown in Figure 7. Once the medical image is uploaded to the clinical data repository, a *Data table* entry will be created in the database. Before storing the file on the clinical data repository file system, it is renamed to a unique combination of characters and numbers. The *Data table* entry stores the original filename, the new filename and the size of the file. The system will calculate the hash code of each file by calling a hash function, which is used to map data of arbitrary size to data of fixed size. This function provides a unique identifier to the file, which can be used to ensure the consistency of the data. The same file will always produce the same hash code. This allows the system to recognize if a file is already stored on the file system. If a file being uploaded already exists in the database, the system either rejects the file or creates a symbolic link. The approach helps to maintain the consistency of the data and to optimally use disk space.

User file system:

Name	Änderungsdat...	Typ	Größe
sample.dcm	08.03.2010 18:17	DCM-Datei	550 KB

Database:

<div> <div>Data</div> <div> <div>Properties</div> <div> <div>Id</div> <div>CreatedDate</div> <div>OriginalFileName</div> <div>HashCode</div> <div>AnonymizedHashCode</div> <div>Size</div> <div>Key</div> <div>IsSymLink</div> <div>FSPath</div> <div>FSPathTarget</div> </div> <div>Navigation Properties</div> <div>ObjectVersions</div> </div> </div>	<div> <div>1</div> <div>2014-05-14 16:56:29.770</div> <div>sample.dcm</div> <div>6F7533F66B7C6783CFA7391CB5439A48522B8712</div> <div>A3E3EE841FE0311CD20C92F3315791C7123D2D68</div> <div>563200</div> <div>0</div> <div>False</div> <div>Dicom\0000afc0b2c94eeda1e0e54a690c47c3.vsd</div> <div>NULL</div> </div>
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Clinical data repository file system:

Name	Date modified	Type	Size
0000afc0b2c94eeda1e0e54a690c47c3.vsd	10.06.2011 09:23	VSD File	550 KB

Figure 7: The clinical data repository uses a *Data table* to store the information about the file, which includes the physical location of the file, its size, name, upload time and hash code. This information is used to ensure the consistency of the data stored in this repository.

3.5 Software architecture and design

The clinical data repository is divided into layers based on the multi-tier architecture. The layering model is based on a responsibility layering strategy that associates each layer with a particular responsibility. This strategy has been chosen because it isolates various system responsibilities from one another, so that it improves both system development and maintenance.

3.5.1 Presentation layer

The presentation layer is the topmost level of the clinical data repository. This layer can be accessed directly by users such as administrators, data providers, model providers, researchers, clinicians and software developers. Each type of user will have specific access right to the system. For example, the

administration tools will only be accessible by system administrators. These tools will include an interface to view the application logs and to manage permissions.

The website will be accessible by all users and serves as an interface to view and manage the content of the clinical data repository. The website leverages the Model-View-Controller (MVC) pattern. The MVC pattern separates the representation of information from the user's interactions with the data. The model represents the data and does not depend on the controller or the view. The view displays the model data, and sends user actions (e.g. button clicks) to the controller. The controller provides model data to the view, and interprets user actions such as button clicks. The controller depends on the view and the model.

The REST (Representational State Transfer) API (Application Programming Interface) will provide a complete access to the features of the database including authentication, authorization, data upload, data download, linking etc. The first application to rely on the REST API will be the data upload tool developed by CUSTODIX to provide pseudonomized dataset to the clinical repository. The second application to rely on the REST API will be portlets, which will be integrated into the CHIC portal and will be accessible by all users. Portlets are pluggable user interface software components that are managed and displayed in a web portal. They produce fragments of markup code such as HTML, XHTML or WML.

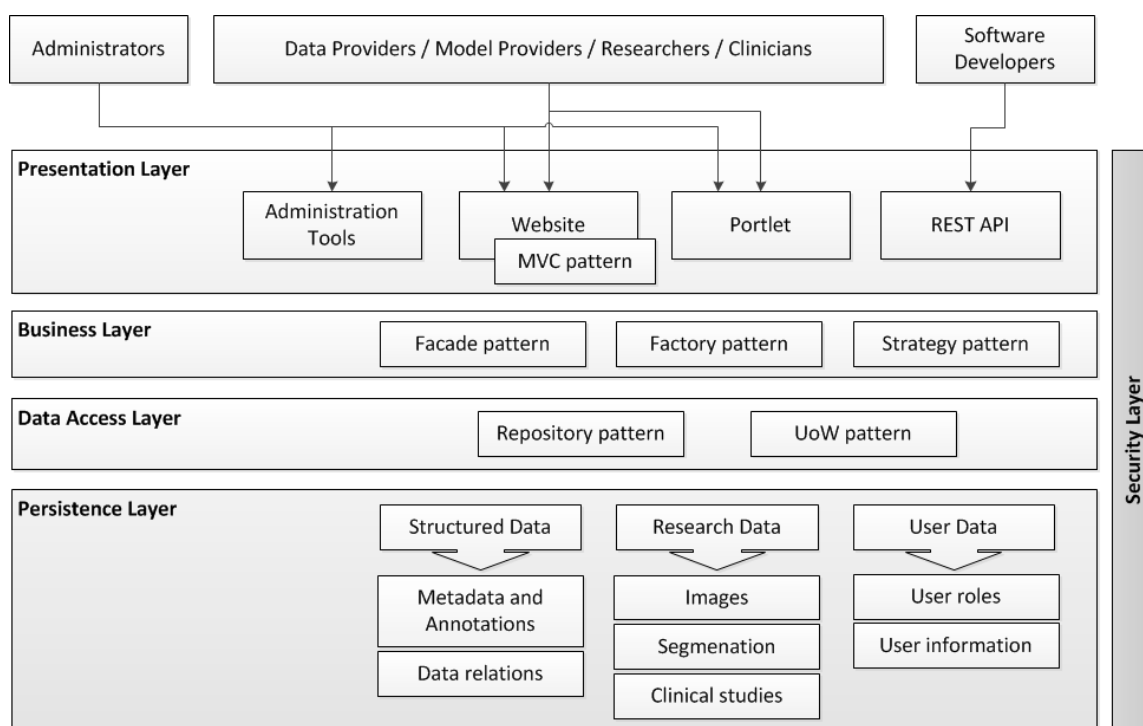


Figure 8: The multi-tier architecture of the clinical data repository

3.5.2 Business layer

The business layer processes commands initiated by the presentation tier, makes logical decisions and evaluations, and performs calculations. Common software design patterns such as the facade pattern, factory pattern and strategy pattern will be used. Software design patterns provide easy to recognize and use object-oriented programming solutions to common problems. They're inherently easy to maintain, because many people are familiar with them.

- The facade pattern [9] is a structural pattern, which simplifies the interface to a complex system. It is composed of all the classes that compose each subsystems of the complex system. A facade shields the user from the complex details of the system and provides them with a simplified view, which is easy to use. It also decouples the code used by the system from the details of the subsystems, making the system easier to modify.
- The factory pattern [9] is a creational pattern that uses factory methods to deal with the problem of creating objects without specifying the exact class that will be created. This is done by creating objects via a factory method, which is either specified in an interface (abstract class) and in the implementing classes (concrete classes) or implemented in a base class (optionally as a template method), which can be overridden when inherited in derived classes (rather than by a constructor).
- The strategy pattern [9] is a behavioural pattern that defines a family of algorithms, encapsulated in another, and makes them interchangeable. The strategy pattern lets the algorithm vary independently from clients that use it.

3.5.3 Data access layer

The data access layer makes use of the repository pattern [10] and Unit of Work (UoW) pattern [11] to mediate between the persistence layer and the business layer of the application. The repository pattern is used to separate the logic that retrieves the data and maps it to the entity model from the

business logic that acts on the model. It queries the data source for the data, maps the data from the data source to a business entity, and persists changes in the business entity to the data source. Three benefits of the repository pattern are:

- It centralizes the data logic and service access logic.
- It provides a substitution point for unit tests.
- It provides a flexible architecture that can be adapted as the overall design of the application evolves.

Besides the repository pattern, the data access tier makes use of the Unit of Work pattern. According to Martin Fowler, the Unit of Work pattern "maintains a list of objects affected by a business transaction and coordinates the writing out of changes and the resolution of concurrency problems" [25]. In other words the pattern takes care of all transaction-handling code. The responsibilities of the pattern are to:

- Manage transactions.
- Order the database inserts, deletes, and updates.
- Prevent duplicate updates.

The value of using the Unit of Work pattern is to free the rest of the code from these concerns so that most of the effort can be spent on business logic.

3.5.4 Persistence layer

The persistence layer is the last instance for data and is represented by a relational database. Cache functionalities will be put in place between the persistence layer and the data access layer to prevent repeated expensive calls to the relational database. The domain model used to describe the structure of the database can be found in chapter 3.6.

3.5.5 Security layer

The security layer is responsible for authenticating and authorizing purposes. This layer is described in chapter 3.8.

3.6 *Domain Model*

A domain model in software engineering is a conceptual model of all the topics related to a specific problem and is helpful during the development process of the database schema. The domain model illustrated in Figure 9 represents the vocabulary and key concepts of the problem domain. It also identifies the relationships among all the entities within the scope of the problem domain. A problem domain in this particular case is the application that needs to be examined to solve the problem of storing clinical data. Two different kinds of entities are used within the domain model. On one hand the domain classes (blue) and on the other hand the domain enumerations (brown). Domain classes represent the data to be stored and domain enumerations represent constant data which are persisted once. Domain enumerations cannot be altered by the users of the application. Figure 9 describes the information stored in the database. For each object version the information includes authentication/authorization, metadata and annotations as well as specific instructions concerning the uploaded file and the owner of the data. A more precise description of each table is given below.

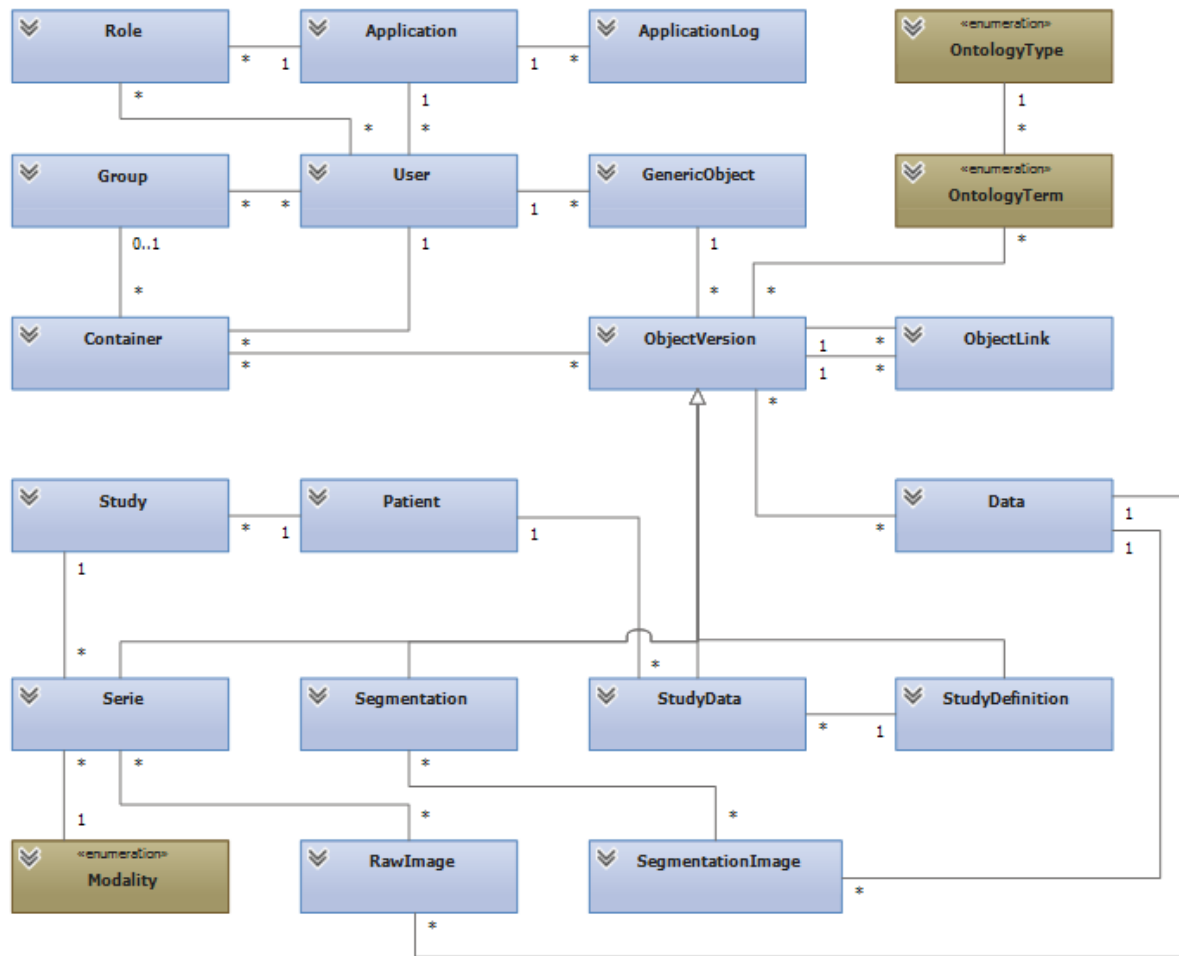


Figure 9: The domain model of the clinical data repository with domain classes (blue), domain enumerations (brown) and their relationships represented as connecting lines. A brief description of each entry is given in the table below.

Table 1: Domain class description of the clinical data repository.

Name	Description
Role	Roles are used for authorization purposes. Possible roles are user, administrator, clinician, etc. A user can have multiple roles and a role is bound to an application.
Application	The application entity is used to support running multiple instances of the software with a different user base and roles.

ApplicationLog	ApplicationLog entries store exception messages raised by the health monitoring system.
Group	Groups are used for organization and authorization purposes. System groups can be created by administrators only and virtual groups can be created by all users. A group can have multiple users and a user can have multiple groups.
User	A user entity contains personal information of the user such as the first name and last name. The email address is used to identify the user and must be application-wide unique. A user can be locked out for a certain time window or deactivated permanently.
Container	A container represents a folder in the virtual file system. A container can contain other containers or data objects.
ObjectVersion	The data object (ObjectVersion) is the basis element of the clinical data repository.
GenericObject	GenericObjects are used for versioning purposes. A new version of a data object (ObjectVersion) will have the same GenericObject as the ancestor.
OntologyType	An OntologyType could be for example the Foundational Model of Anatomy (FMA) or other ontologies which are being developed within the CHIC project.
OntologyTerm	An OntologyTerm is an entry of the respective OntologyType.
ObjectLink	An ObjectLink is used to link a data object with another.
Study	The name of this entity refers to the study (sometimes referred to as exam or procedure) of the DICOM model. A study can have multiple series and a patient can have multiple studies but the study has exactly one patient.

Patient	A Patient is a person receiving, or registered to receive, healthcare services, or is the subject of one or more studies for some other purpose, such as research.
Data	The Data entity contains the information where exactly one binary file is stored on the clinical data repository file system. A Data entity can be used by multiple ObjectVersions and an ObjectVersion can have multiple Data entities.
Serie	A Serie data object represents one or multiple raw images.
Segmentation	A Segmentation data object represents one or multiple segmentation images.
StudyData	A StudyData data object represents the captured Patient data of a single clinical study.
StudyDefinition	A StudyDefinition data object represents the definition of a single clinical study.
Modality	Medical imaging modalities are for example CT (Computed Tomography), CR (Computed Radiography), US (Ultrasound), etc.
RawImage	A RawImage entity contains the information where exactly one extracted thumbnail image is stored on the clinical data repository file system.
SegmentationImage	A SegmentationImage entity contains the information where exactly one extracted image and thumbnail is stored on the clinical data repository file system.

3.7 Technology

The development environment of the clinical data repository is mainly based on Microsoft technologies. The system relies on a MS SQL (Microsoft Structured Query Language) database and an IIS web-server (Microsoft Internet Information Service). The programming language of the core

application is C# in combination with the .NET Framework. The website makes use of HTML, CSS and JavaScript. Third-party libraries, frameworks and technologies used by the clinical data repository are listed in Table 2.

Table 2: Third-party libraries, frameworks and technologies used by the clinical data repository.

Name	Description
Entity Framework	The Entity Framework is an open source object-relational mapping (ORM) framework for ADO.NET, part of .NET Framework.
ClearCanvas	ClearCanvas is an open source library used to read DICOM files.
ITK	Insight Segmentation and Registration Toolkit is an open source library used to read Metalmage, Analyze and Niftii files.
Fuseki	Fuseki is a SPARQL server using HTTP which enables to query external services such as a folksonomy semantic service.
ASP.NET Web API	The framework used to build the REST API.
ASP.NET MVC	The ASP.NET MVC is an open source web application framework that implements the model–view–controller (MVC) pattern.
Bootstrap	Bootstrap is a free collection of tools for creating websites. It contains HTML and CSS-based design templates and optional JavaScript extensions.
jQuery	jQuery is a cross-platform JavaScript library designed to simplify the client-side scripting of HTML.

3.8 Security

The clinical data repository leverages the security system of the .NET Framework in combination with the CHIC security framework. Two concepts called role-based security and object-based security are distinguished for authorization purposes. The clinical data repository itself will act as a Service

Provider (SP) within the CHIC security system. Security tokens issued by trusted Identity Providers (IdP) will be mapped accordingly in order to support single sign-on (SSO).

3.8.1 Role-based security

The role-based security [12] in the .NET Framework supports authorization by making information about the principal, which is constructed from an associated identity, available to the current thread. A principal represents the identity and role of a user and acts on the user's behalf. Role-based security in the .NET Framework supports three kinds of principals:

- Generic principals represent users and roles that exist independently of Windows users and roles.
- Windows principals represent Windows users and their roles (or their Windows groups). A Windows principal can impersonate another user, which means that the principal can access a resource on a user's behalf while presenting the identity that belongs to that user.
- Custom principals can be defined by an application in any way that is needed for that particular application. They can extend the basic notion of the principal's identity and roles.

The clinical data repository uses custom principals for the role-based security. The identity (and the principal it helps to define) can be either based on a Windows account or be a custom identity unrelated to a Windows account. Applications built with the .NET Framework can make authorization decisions based on the principal's identity or role membership, or both. A role is a named set of principals that have the same privileges with respect to security. A principal can be a member of one or more roles. Therefore, applications can use role membership to determine whether a principal is authorized to perform a requested action.

3.8.2 Object-based security

The object-based security applies to individual instances of entities and is provided by using access rights. An access right is granted to a user or a group for each particular entity instance. The relationship between an access right and a role is that access rights apply only after roles have taken effect. For example, if users are not in the role to read objects, they will be unable to read any object,

regardless of the access rights another user might grant them to a specific account through sharing. Unlike role-based security that is natively implemented within the .NET Framework, the object-based security needs to be developed. Figure 10 illustrates how object-based security will be implemented in the clinical data repository.

Table 3: Object-based security domain class description of the clinical data repository.

<i>Name</i>	<i>Description</i>
Right	Possible rights could be for example none, visit, read, download, edit, manage, owner, etc.
ObjectRight	ObjectRights can be assigned to ObjectVersions.
ContainerRight	ContainerRights can be assigned to Containers.
RightSet	A RightSet is a predefined set of rights. Possible RightSets could be for example protect, default, collaborate, private, etc.
ObjectRightSet	An ObjectRightSet is a predefined set of ObjectRights.
ContainerRightSet	A ContainerRightSet is a predefined set of ContainerRights.
UserRight	A UserRight is a right assigned to a user.
ObjectUserRight	An ObjectUserRight is a right assigned to an ObjectVersion and a User.
ContainerUserRight	A ContainerUserRight is a right assigned to a Container and a User.
GroupRight	A GroupRight is a right assigned to a group.
ObjectGroupRight	An ObjectGroupRight is a right assigned to an ObjectVersion and a Group.
ContainerGroupRight	A ContainerGroupRight is a right assigned to a Container and a Group.

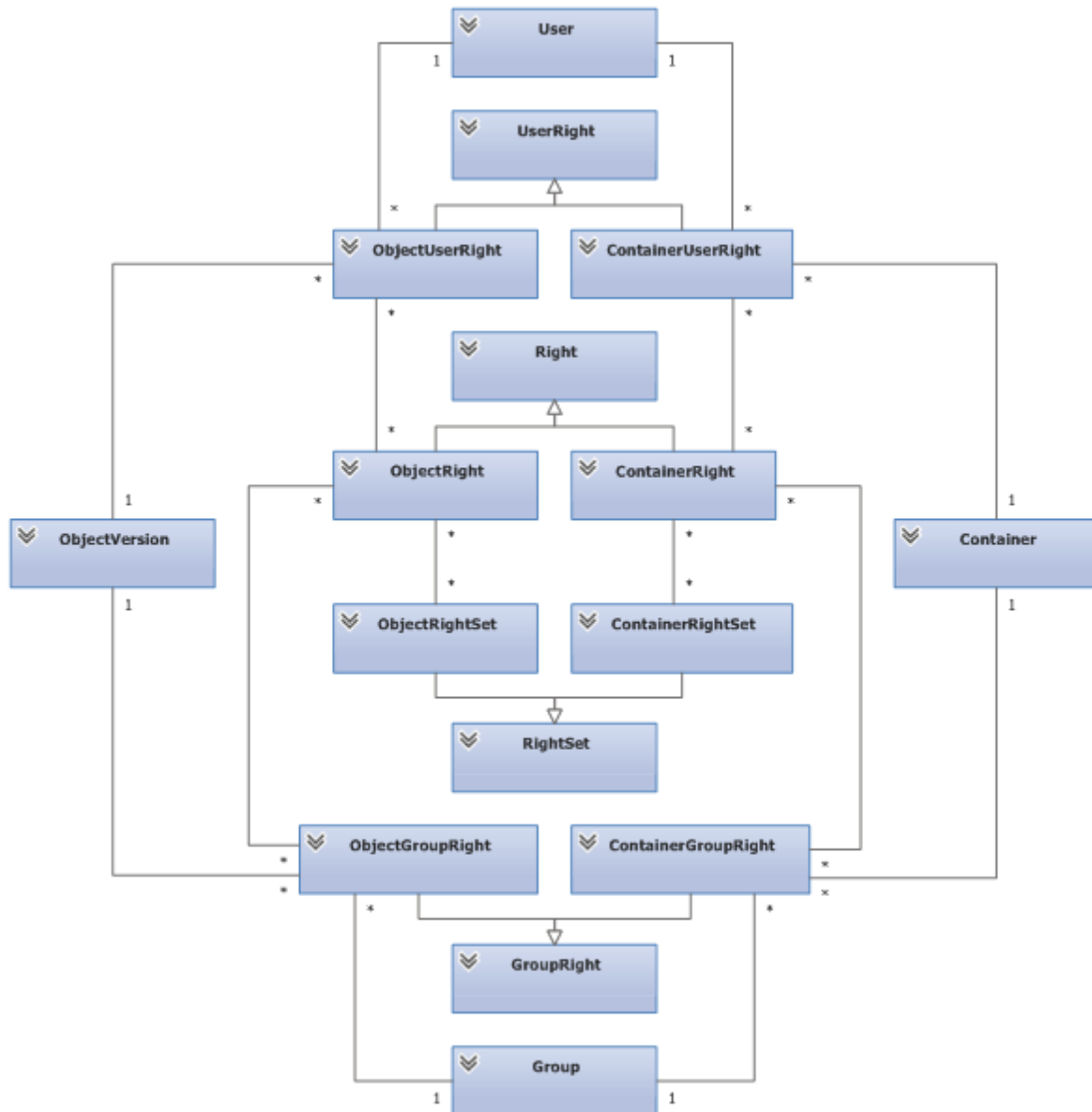


Figure 10: Object-based security in the clinical data repository.

3.9 Data representation

This chapter gives an overview graphical user interface (GUI) of the clinical data repository. The concept is described based on mock-ups which will serve as foundation for the final implementation with the chosen web technology.

On the top left corner the virtual folder structure of the authenticated user will be displayed (1). The virtual folder structure is described in chapter 3.4.4. Below, the page navigation provides shortcuts to navigate to the selected section on the current page quickly (2). On the top middle of Figure 11 an

input field will be placed with the functionality to search the clinical data repository (3). The supporting toolbar on top of the list can be used to perform actions on the selected items (4). Those actions include modifying the metadata, changing the permissions, creating links, initiating the download etc. It is also possible to perform those actions on each item individually (5). Each item can be rated for quality and tagged with personal comments (6). Each item in the list is uniquely named based on a predefined template (7) whereat a list item is a data object as described in chapter 3.4.1. Buttons to collapse/expand the respective information of the item will be placed to the right of the item name (8). At the bottom of Figure 11 the pagination will be displayed which divides the content into discrete pages (9).

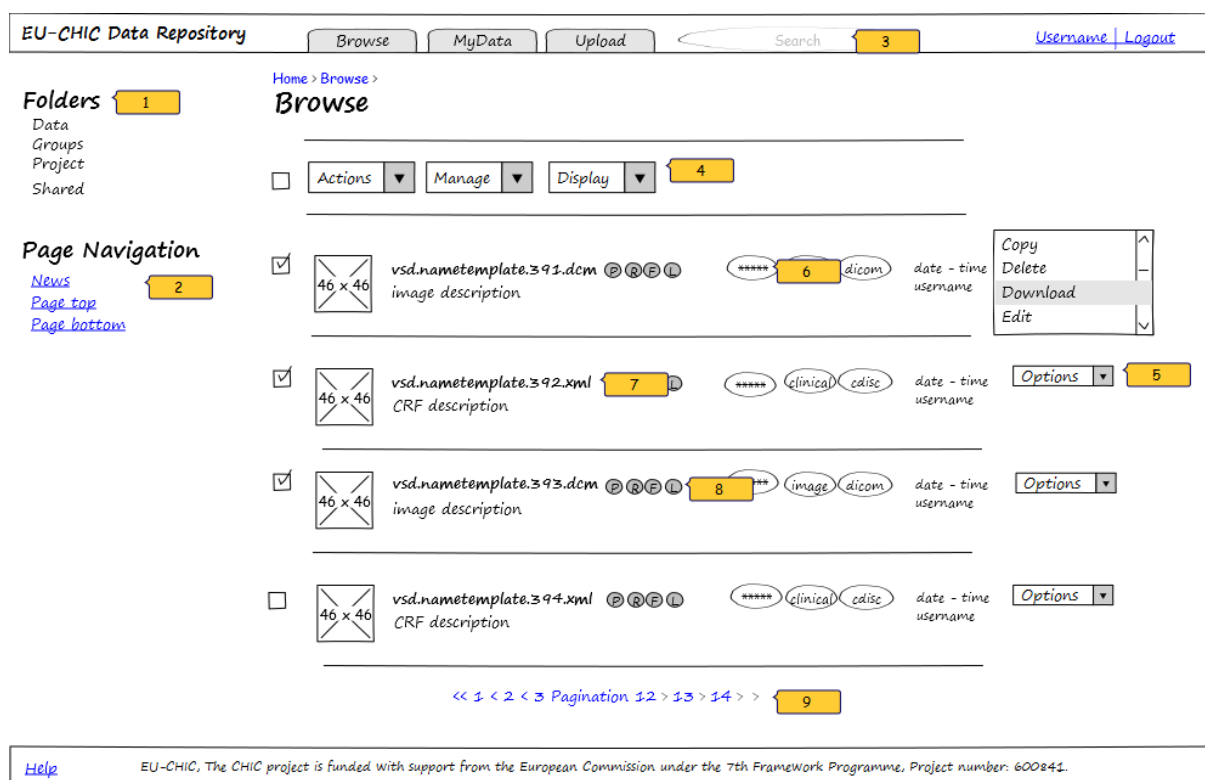


Figure 11: Clinical data repository mock-up of the data list view.

The detail view of an item can be reached by clicking on the item name in the list view. The general information about the item can be changed (1). The section in the middle of Figure 12 shows the related data for the current object (2). Which user and/or which group is allowed to access the item is shown in the permissions section (3). Older and/or newer versions of the current item are listed in the last section (4).

EU-CHIC Data Repository

[Browse](#) [MyData](#) [Upload](#) [Username](#) | [Logout](#)

Home > [Browse](#)

Folders

- Data
- Groups
- Project
- Shared

Page Navigation

- [File Infos](#)
- [Previews](#)
- [Related data](#)
- [Permissions](#)
- [Filehistory](#)
- [Page top](#)
- [Page bottom](#)

Details vsd.nametemplate.391.dcm

[Actions](#) [Download](#) [Copy](#)

File information [Edit](#) 1

Description

Anatomical Regions

Type

Liver

Liver cell

Liver stroma

Liver parenchym

Gender

Modality

Preview

46 x 46

46 x 46

46 x 46

46 x 46

46 x 46

46 x 46

46 x 46

46 x 46

46 x 46

46 x 46

Related Data

<input type="checkbox"/>	file	
<input type="checkbox"/>	vsd.anyfile.129.dcm	Download
<input type="checkbox"/>	vsd.crf.123.xml	Download
<input type="checkbox"/>	vsd.anyct.394.dcm	Download

Permissions

<input type="checkbox"/>	perm.	group / user	
<input type="checkbox"/>	download	chic-eu	change
<input type="checkbox"/>	visible	registered	change
<input type="checkbox"/>	private	world	change

Versions

<input type="checkbox"/>	file	
<input type="checkbox"/>	vsd.anyfile.122.dcm	Download
<input type="checkbox"/>	vsd.crf.003.dcm	Download
<input type="checkbox"/>	vsd.anyct.001.dcm	Download

EU-CHIC, The CHIC project is funded with support from the European Commission under the 7th Framework Programme, Project number: 600841.

Figure 12: Clinical data repository mock-up of the data detail view.

4 Model/Tool Repository and *In Silico* Trial Repository

4.1 Introduction

The model/tool repository is the technological platform where hypomodels, hypermodels, data transformation tools, linkers and other tools used in the construction and execution of hypermodels are stored. End users and other technological components may access the stored models/tools, or store new ones, using the graphical or programmatic interfaces provided by the model/tool repository.

The *in silico* trial repository is the technological platform where the data related to *in silico* trials and *in silico* experiments are stored. The design of the *in silico* trial repository is based on the classical and standardized principles of real clinical trials. Similar to the model/tool repository, end users and other technological components may access the *in silico* trial repository in order to prepare new *in silico* trials or to store the results of the conducted *in silico* experiments.

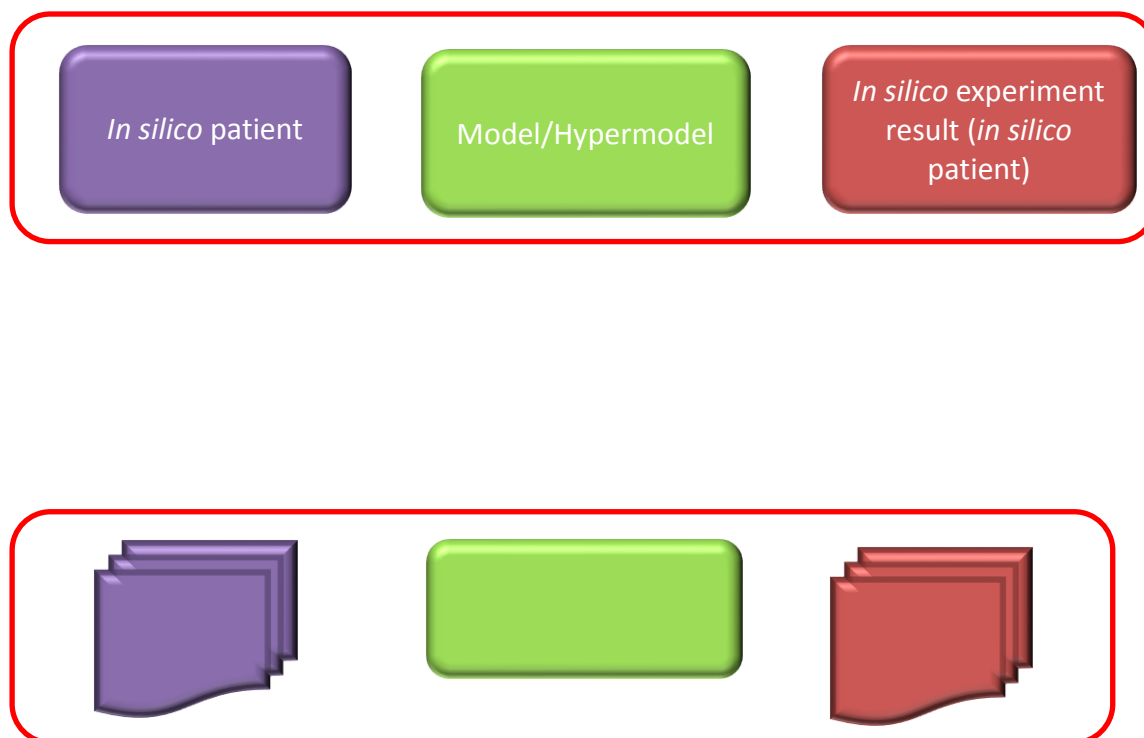
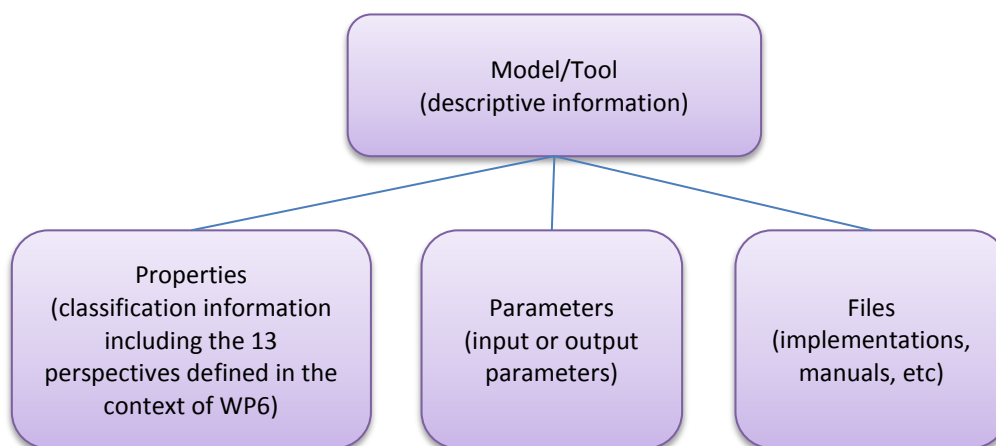


Figure 14: *In silico* trial: Group of *in silico* experiments in which the same model/hypermodel has been used.

The following paragraphs will describe the information model used by the model/tool repository and the *in silico* trial repository. The graphical and programmatic interfaces of the aforementioned repositories will be reported in “D8.2: *Prototype implementation of the CHIC repositories*”, “D8.3: *Implementation of the interfaces of the CHIC repositories*” and “D10.2: *Design of the orchestration platform, related components and interfaces*”.

4.2 Model/Tool Repository Data Layer

The model/tool repository consists of four basic entities: the models/tools, the properties, the parameters and the files.



The basic principles of the model/tool repository are:

- Each model/tool has basic descriptive information, stored in the entity “mr_tool”. This information uniquely defines the model/tool and differentiates it from other models/tools.
- Each model/tool can have one or more properties that further describes or/and classifies it. It must be noted that properties can only be used in correlation with “mr_tool” entity as they may only supplement the basic descriptive information of a model/tool. In CHIC project there

is a set of predefined properties that reflect the 13 perspectives defined in “D6.1: *Cancer hypomodelling and hypermodelling strategies and initial component models*”.

- The descriptive information of properties is stored in the entity “mr_property”. This entity does not contain the value of the property (related to a specific model/tool), but only the description of the property. The value that a property takes in case of a specific model/tool is stored in the entity “mr_tool_property”.
- The entities “mr_tool” and “mr_property” assist the user in understanding the nature of the model/tool and facilitate the abstract construction of hypermodels. However, the ultimate goal of CHIC is to provide concrete specifications of models/tools that can be joined together programmatically. In order to succeed this, we associate each model/tool with a set of parameters.
- The models are treated as generic stubs, as described in “D7.1: *Hypermodelling specifications*”, which have entry and exit points. Consequently, each model/tool has various parameters, serving as input parameters or output parameters, which are stored in entity “mr_parameter”. This entity facilitates the transition from an abstract representation to a concrete one. Logical compatibilities between connected parameters must be taken into account along with the aspect of units, in order to avoid inconsistencies between the connected models/tools.
- Each tool may be associated with a set of references, stored in the entity “mr_reference”, which provide direct or indirect links to additional material, extending in this way the knowledge base related to the specific model/tool.
- Every model/tool can be accompanied by a set of files. The information concerning the aforementioned files is stored in the entity “mr_file”. The entity “mr_file” only holds the metadata of the file and not its data. The data of the files are stored internally in a file based repository. If a file is an implementation or a computational representation of a model/tool, then a suitable engine is specified for running the file.

Entity: mr_tool

- *id*: Primary key. Used to uniquely identify each table row.
- *title*: The name of the model/tool.
- *description*: The (short) textual description of the model/tool.
- *comment*: Any comment that the creator/modifier of the model/tool wants to include.
- *created_on*: The date and time that the record has been created.
- *created_by*: The id of the creator of the record.
- *modified_on*: The date and time that the record has been modified.
- *modified_by*: The id of the modifier of the record.

Entity: *mr_reference*

- *id*: Primary key. Used to uniquely identify each table row.
- *tool_id*: Reference. Linked to the model/tool that this resource refers to. Linked to the entity “mr_tool”.
- *title*: The name given to the resource.
- *type*: The type of the resource. Example values: “book”, “journal article” etc.
- *creator*: The creator(s) of the resource (e.g. authors etc.).
- *issued*: The date of formal issuance (e.g. publication) of the resource.
- *bibliographic_citation*: The bibliographic citation of the resource.
- *is_part_of*: The related resource that this resource is part of.
- *source*: The related resource from which the described resource is derived from.
- *doi*: The DOI (Digital Object Identifier) of the resource. This field is empty if the resource doesn’t have a DOI.

- *pmid*: The PubMed identifier. This field is empty if the resource is not included in the PubMed database.
- *created_on*: The date and time that this record has been created.
- *created_by*: The id of the creator of this record.
- *modified_on*: The date and time that this record has been modified.
- *modified_by*: The id of the modifier of this record.

Entity: *mr_property*

- *id*: Primary key. Used to uniquely identify each table row.
- *name* : The name of the property.
- *description*: The (short) textual description of what this property represents.
- *comment*: Any comment that the creator of the property wants to include.
- *semtype*: A url representing semantic information about this property.

Entity: *mr_tool_property*

- *id*: Primary key. Used to uniquely identify each table row.
- *tool_id*: Reference. The id of the model/tool. Linked to the entity “mr_tool”.
- *property_id*: Reference. The id of the property. Linked to the entity “mr_property”.
- *value*: The value that the property (*property_id*) takes in case of a specific model/tool (*tool_id*).
- *created_on*: The date and time that this record has been created.
- *created_by*: The id of the creator of this record.

- *modified_on*: The date and time that this record has been modified.
- *modified_by*: The id of the modifier of this record.

Entity: *mr_parameter*

- *id*: Primary key. Used to uniquely identify each table row.
- *tool_id*: Reference. The id of the model/tool that this parameter is associated with. Linked to the entity “mr_tool”.
- *name* : The name of the parameter.
- *description*: The (short) textual description of what this parameter represents.
- *data_type*: The type of the parameter. Possible values can be “number”, “string” or “file”.
- *unit*: The units in which the parameter is represented. Only applicable if the parameter is a number.
- *data_range*: The range of the parameter values separated by “-”. Only applicable if the parameter is a number.
- *default_value*: The value that will be used if a parameter value is not provided to the tool.
- *is_mandatory*: True if this is a mandatory parameter.
- *is_output*: True if this parameter is an output parameter.
- *comment*: Any additional comment concerning this parameter.
- *semtype*: A url representing semantic information about this parameter.
- *created_on*: The date and time that this record has been created.
- *created_by*: The id of the creator of this record.
- *modified_on*: The date and time that this record has been modified.

- *modified_by*: The id of the modifier of this record.

Entity: *mr_file*

- *id*: Primary key. Used to uniquely identify each table row.
- *tool_id*: Reference. Linked to the model/tool that this file is associated with. Linked to the entity “mr_tool”.
- *title*: The name of the file.
- *description*: The (short) textual description of what this file represents.
- *kind*: Defines what this file is. Example values: “document”, “source code”, “binary” etc.
- *source*: The location where this file is internally stored.
- *version*: The version of the file. Multiple versions of the same file can be stored in the repository.
- *license*: The license associated with this file. It can be the name of a well-known license (Apache, MIT etc) or the detailed description of the license.
- *sha1sum*: The sha1 checksum of this file (data). It is used in order to check the consistency of the file.
- *comment*: Any additional comment.
- *engine*: The engine that is suitable for executing this file. Only applicable in case that the file can be executed/run.
- *created_on*: The date and time that this record has been created.
- *created_by*: The id of the creator of this record.
- *modified_on*: The date and time that this record has been modified.

- *modified_by*: The id of the modifier of this record.

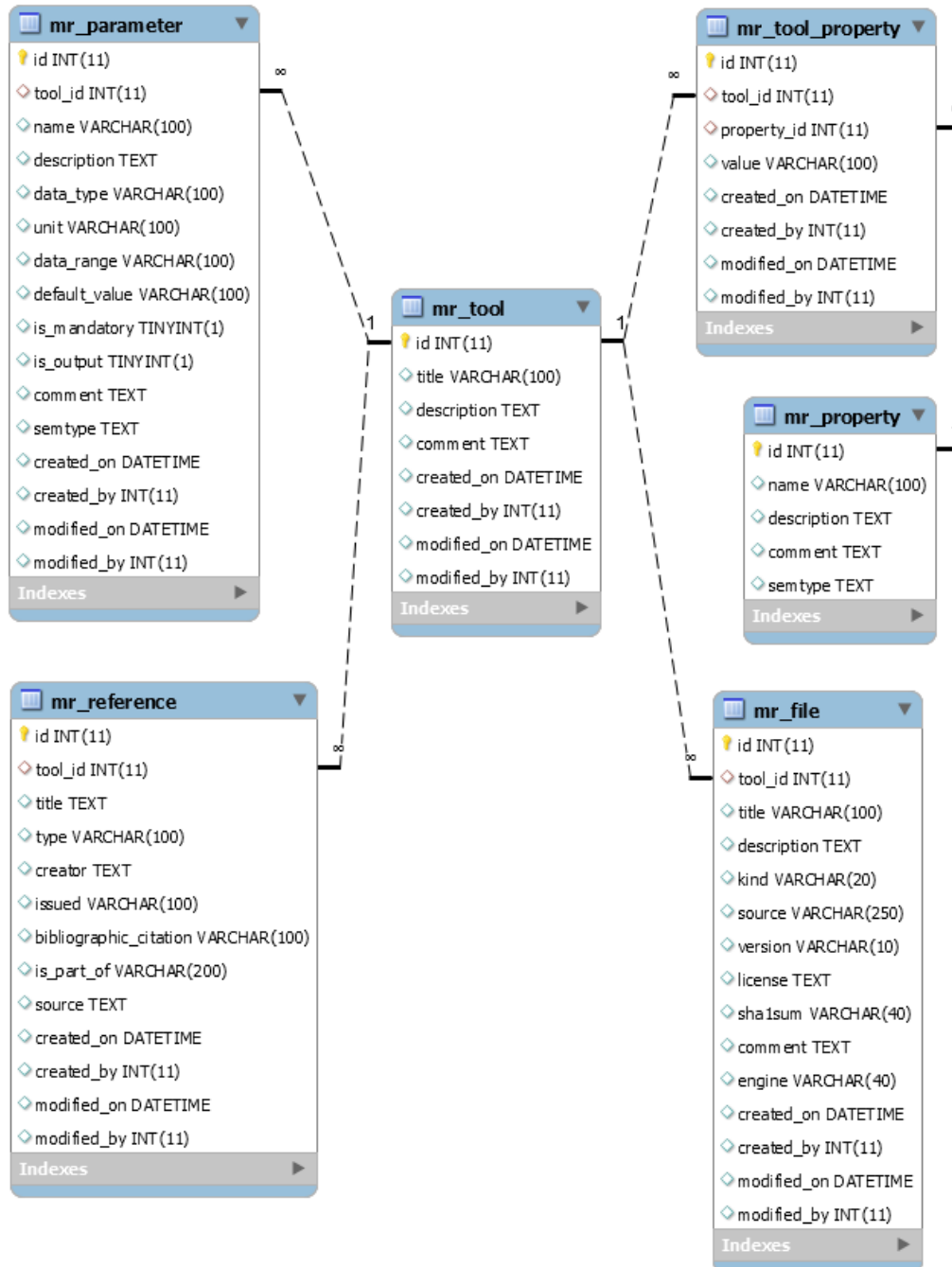


Figure 16: Entity-Relationship (ER) diagram of model/tool repository.

4.3 Repeatability and Reproducibility

One of the main purposes of the *in silico* trial repository is to test the repeatability and reproducibility of the experiments conducted in the context of *in silico* cancer domain.

Repeatability (see Figure 17) is the ability for an individual to show that an experiment, repeated using the same material and equipment, yields the same result. In *in silico* medicine this means that if we run the same module multiple times on the same computer using the same software the same result would be yielded. For a stochastic simulation, multiple runs on the same computer will yield the same statistical distribution.

Repeatability

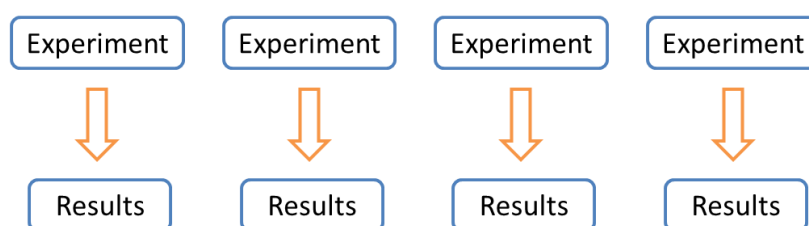


Figure 17: Repeatability

Reproducibility (see Figure 18) is the ability for different individuals to show that an experiment repeated using different but similar material and different equipment yields the same statistical result. In *in silico* medicine this means that we are able to recreate a simulation without necessarily using the same software or computer that was used in the original simulation. Reproducing an experiment is one important approach that scientists use to gain confidence in their conclusions.

Recently, the scientific community was shaken by reports that a troubling proportion of peer-reviewed preclinical studies are not reproducible [26]. Because confidence in results is of paramount importance to the broad scientific community, new initiatives are announced to increase confidence in the studies.

Reproducibility

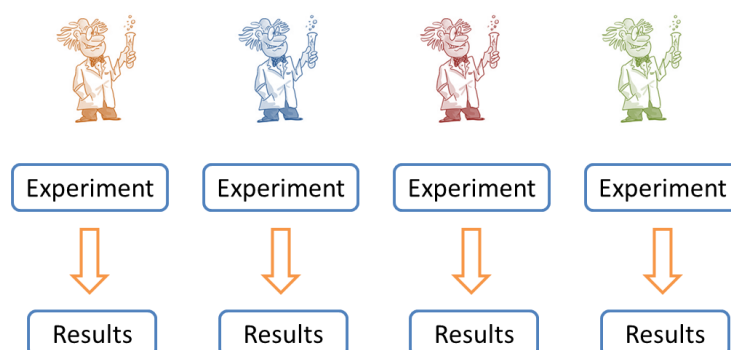


Figure 18: Reproducibility

The *in silico* trial repository can serve perfectly the aforementioned initiatives. By storing in one place the complete information concerning the input data, the output data and the modules which participate in the *in silico* experiments and the *in silico* trials, the *in silico* trial repository can advance *in silico* medicine in general, by facilitating the validation of the current *in silico* medicine discoveries.

4.4 *In Silico* Trial Repository Data Layer

The *in silico* trial repository consists of three main entities, the subjects, the experiments and the trials.

The basic principles of the *in silico* trial repository are:

- The subject entity actually represents an instance of a subject. The subject may be a person, healthy or not, an animal etc. The subject can be linked to another data repository, such as the CHIC clinical data repository, a clinical trial management system (ObTiMA [18], OpenClinica [19] etc), a hospital record management system etc. Every instance of a subject can be accompanied by a set of files. The data of the files are stored internally in a file based repository.
- The *in silico* experiment entity consists of triples of “initial state of the subject” - “*in silico* (hyper)model” – “final state of the subject”. The *in silico* (hyper)model that is used in an *in*

in silico experiment is not stated in the experiment entity, but in the *in silico* trial entity in which the experiment belongs.

- The *in silico* experiments are organized in *in silico* trials. All *in silico* experiments that are part of the same *in silico* trial use the same *in silico* (hyper)model.
- In a real clinical trial the hypotheses and the scientific method are tested. In case of *in silico* trials, the *in silico* (hyper)model takes the place of the hypotheses and the scientific method. Consequently, the (hyper)model that is being used (and the location where it is stored) is defined in the *in silico* trial entity.
- To be in alliance with the real clinical trial the term “placebo model” is introduced. In case of cancer disease the placebo model can be a free growth model.

Toward the end of the project we aspire to include in CHIC experiments that perform Optimization, Time Course, Sensitivity Analysis etc.

Entity: *tr_subject*

- *id*: Primary key. Used to uniquely identify each table row.
- *description*: The (short) textual description of the state of the subject. The subject may be a person (healthy or patient), an animal etc.
- *subject_external_id*: The external id of the subject. This field is used only in the case in which this subject’s state (real or virtual) is directly or indirectly linked to a subject stored in an external repository.
- *external_url*: The URL of the external repository mentioned above. Such external repositories can be the CHIC clinical data repository, a clinical trial management system (ObTiMA, OpenClinica etc), a hospital record management system etc.
- *comment*: Any additional comment.
- *created_on*: The date and time that this record has been created.
- *created_by*: The id of the creator of this record.

- *modified_on*: The date and time that this record has been modified.
- *modified_by*: The id of the modifier of this record.

Entity: *tr_trial*

- *id*: Primary key. Used to uniquely identify each table row.
- *description*: The (short) textual description of the experiment.
- *model_id*: The id of the *in silico* model that is used in the trial.
- *model_url*: The URL where the *in silico* model is located. This URL may point to the CHIC model/tool repository or to an external model repository (e.g. the biomodels repository).
- *placebo_model_id*: The *in silico* model that is used as a placebo. Usually in cancer disease this is a free growth model.
- *placebo_model_url*: The URL where the placebo *in silico* model is located.
- *comment*: Any additional comment.
- *created_on*: The date and time that this record has been created.
- *created_by*: The id of the creator of this record.
- *modified_on*: The date and time that this record has been modified.
- *modified_by*: The id of the modifier of this record.

Entity: *tr_experiment*

- *id*: Primary key. Used to uniquely identify each table row.
- *trial_id*: Reference. The id of the trial. Linked to the table “tr_trial”.
- *description*: The (short) textual description of the *in silico* experiment.

- *subject_id_in*: The id of (the state of) subject that is used as an input to the *in silico* experiment.
- *subject_id_out*: The id of (the state of) the subject that is produced after the execution of the *in silico* experiment.
- *placebo*: True if in the *in silico* experiment the “placebo” model must be used.
- *status*: The status of the *in silico* experiment. It can be “NOT STARTED”, “ON PROGRESS”, “FINISHED SUCCESSFULLY”, “FINISHED ERRONEOUSLY” etc.
- *comment*: Any additional comment.
- *created_on*: The date and time that this record has been created.
- *created_by*: The id of the creator of this record.
- *modified_on*: The date and time that this record has been modified.
- *modified_by*: The id of the modifier of this record.

Entity: *tr_reference*

- *id*: Primary key. Used to uniquely identify each table row.
- *title*: The name given to the resource.
- *type*: The type of the resource. Example values: “book”, “journal article” etc.
- *creator*: The creator(s) of the resource (e.g. authors etc).
- *issued*: The date of formal issuance (e.g. publication) of the resource.
- *bibliographic_citation*: The bibliographic citation of the resource.
- *is_part_of*: The related resource that this resource is part of.
- *source*: The related resource from which the described resource is derived from.

- *doi*: The DOI (Digital Object Identifier) of the resource. This field is empty if the resource doesn't have a DOI.
- *pmid*: The PubMed identifier. This field is empty if the resource is not included in the PubMed database.
- *created_on*: The date and time that this record has been created.
- *created_by*: The id of the creator of this record.
- *modified_on*: The date and time that this record has been modified.
- *modified_by*: The id of the modifier of this record.

Entity: *tr_trial_reference*

- *id*: Primary key. Used to uniquely identify each table row.
- *trial_id*: Reference. The id of the trial. Linked to the table “tr_trial”.
- *reference_id*: Reference. The id of the reference. Linked to the table “tr_reference”.
- *created_on*: The date and time that this record has been created.
- *created_by*: The id of the creator of this record.
- *modified_on*: The date and time that this record has been modified.
- *modified_by*: The id of the modifier of this record.

Entity: *tr_experiment_reference*

- *id*: Primary key. Used to uniquely identify each table row.
- *experiment_id*: Reference. The id of the experiment. Linked to the table “tr_experiment”.
- *reference_id*: Reference. The id of the reference. Linked to the table “tr_reference”.

- *created_on*: The date and time that this record has been created.
- *created_by*: The id of the creator of this record.
- *modified_on*: The date and time that this record has been modified.
- *modified_by*: The id of the modifier of this record.

Entity: *tr_file*

- *id*: Primary key. Used to uniquely identify each table row.
- *subject_id*: Reference. Linked to the subject that this file is associated with. Linked to the table “tr_subject”.
- *title*: The name of the file.
- *description*: The (short) textual description of what this file represents.
- *kind*: Defines what this file is. Example values: “document”, “spreadsheet”, “csv”, etc.
- *source*: The location where this file is internally stored.
- *version*: The version of the file. Multiple versions of the same file can be stored in the repository.
- *sha1sum*: The sha1 checksum of this file (data). It is used in order to check the consistency of the file.
- *comment*: Any additional comment.
- *created_on*: The date and time that this record has been created.
- *created_by*: The id of the creator of this record.
- *modified_on*: The date and time that this record has been modified.
- *modified_by*: The id of the modifier of this record.

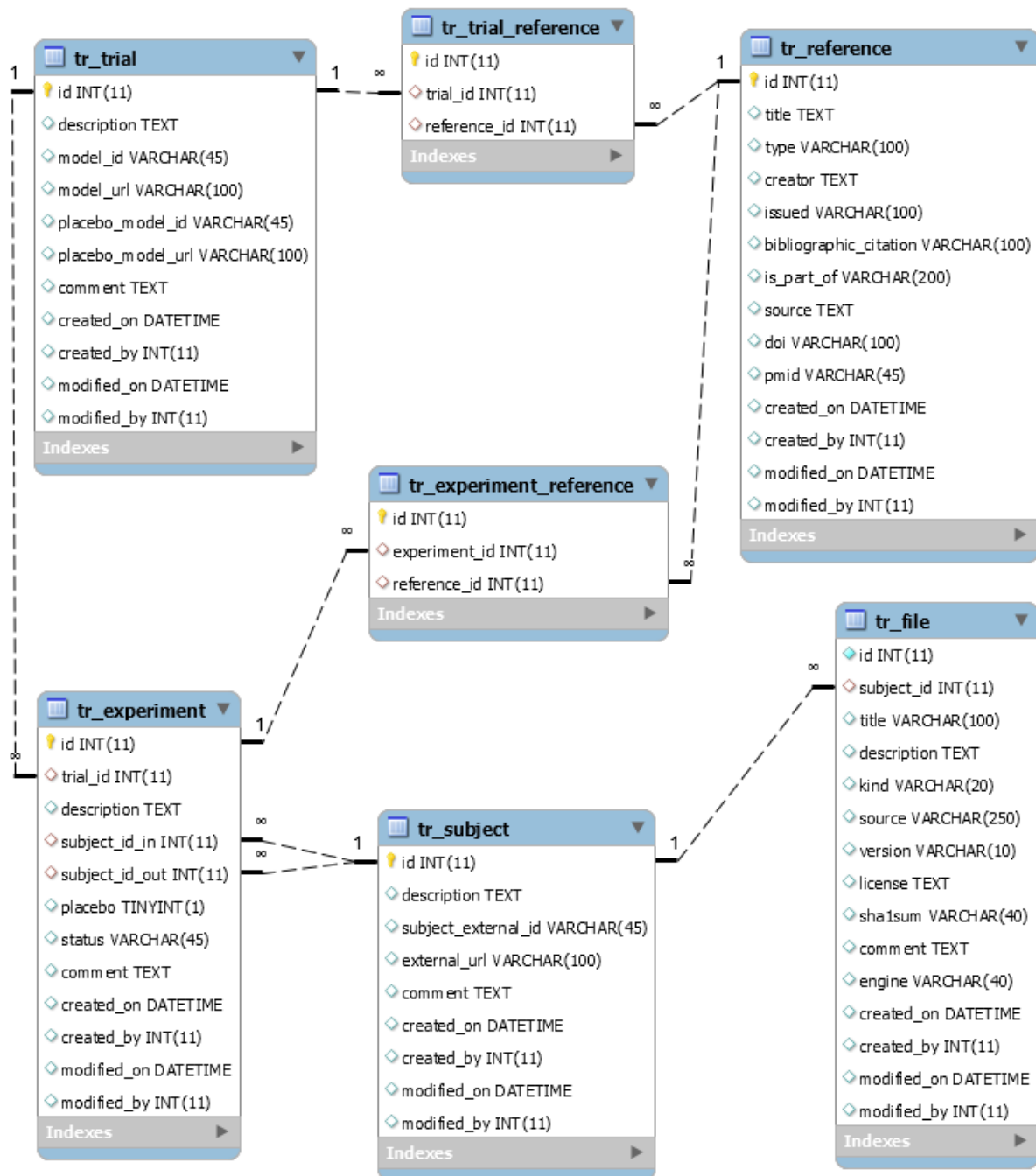


Figure 19: Entity-Relationship (ER) diagram of *in silico* trial repository.

4.5 Authentication and authorization

The model/tool repository and the *in silico* trial repository is built on top of the Django application framework. Consequently, the build-in authentication and authorization mechanism of Django has been adopted and extended appropriately.

The authentication/authorization system consists of a number of parts:

- Users: People registered with the repository.
- Permissions: Binary (yes/no) flags designating whether a user may perform a certain task.
- Groups: A generic way of applying labels and permissions to more than one user.

Users

User objects have the standard username, password, e-mail and real name fields, along with a set of fields that define what the user is allowed to do in the admin interface. First, there is a set of three boolean flags:

- The “active” flag controls whether the user is active at all. If this flag is off and the user tries to log in, he/she won’t be allowed in, even with a valid password.
- The “staff” flag controls whether the user is allowed to log in to the admin interface. Since this same user system can be used to control access to public (i.e., non-admin) domain, this flag differentiates between public users and administrators.
- The “superuser” flag gives the user full access to add, create and delete any item in the admin interface. If a user has this flag set, then all regular permissions (or lack thereof) are ignored for that user.

Groups

Groups are a generic way of categorizing users so that permissions, or other labels, can be applied to those users. A user can belong to any number of groups. A user in a group automatically has the permissions granted to that group. A group is simply a set of permissions to apply to all members of that group. Groups are useful for granting identical permissions to a subset of users.

Permissions

Permissions are used for controlling who has access to create, edit, and delete any object. This permissions system is used to give specific users access only to the portions of the interface that they need. For example, “superusers” have full access to every resource. “Administrators” can create, edit and delete resources. “Common users” can only access resources.

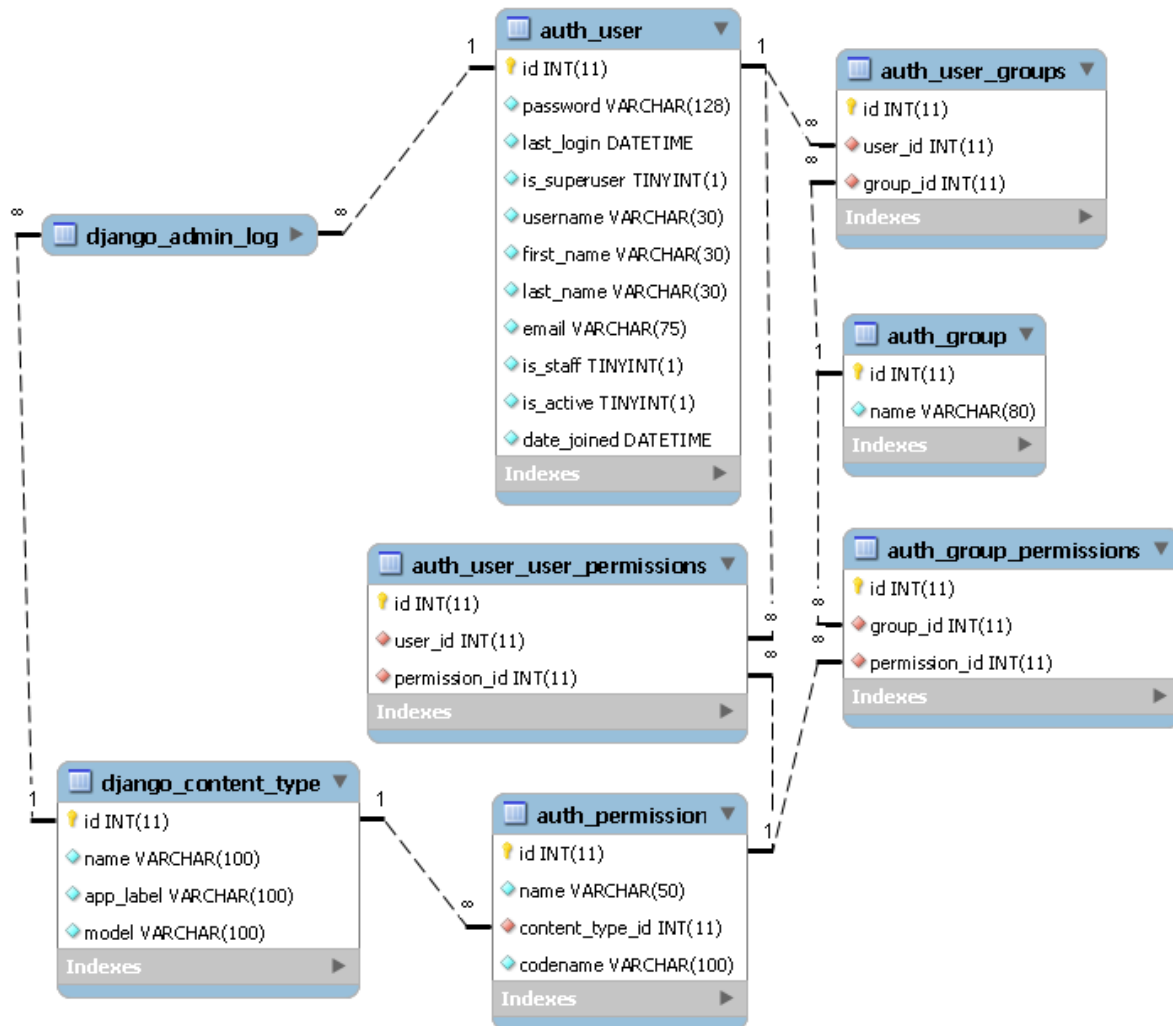


Figure 20: Entity-Relationship (ER) diagram of tables associated with basic authentication and authorization.

In addition to the aforementioned basic authentication and authorization mechanism, the model/tool repository and the *in silico* trial repository will incorporate the single sign on (SSO) mechanism of the CHIC security framework, described in “D5.2: Security guidelines and initial version of security tools”. The integration of the CHIC repositories with the CHIC security framework will be reported in “D8.3: Implementation of the interfaces of the CHIC repositories”.

4.6 Implementation principles and software choices

Before starting the implementation of the model/tool repository and the *in silico* trial repository a thorough investigation was performed of the available software that could be used as a basis. The targeted software had to:

- Be free and open source.
- Have an active community that supports it by building plug-ins and extensions.
- Follow the Model-View-Controller (MVC) paradigm.

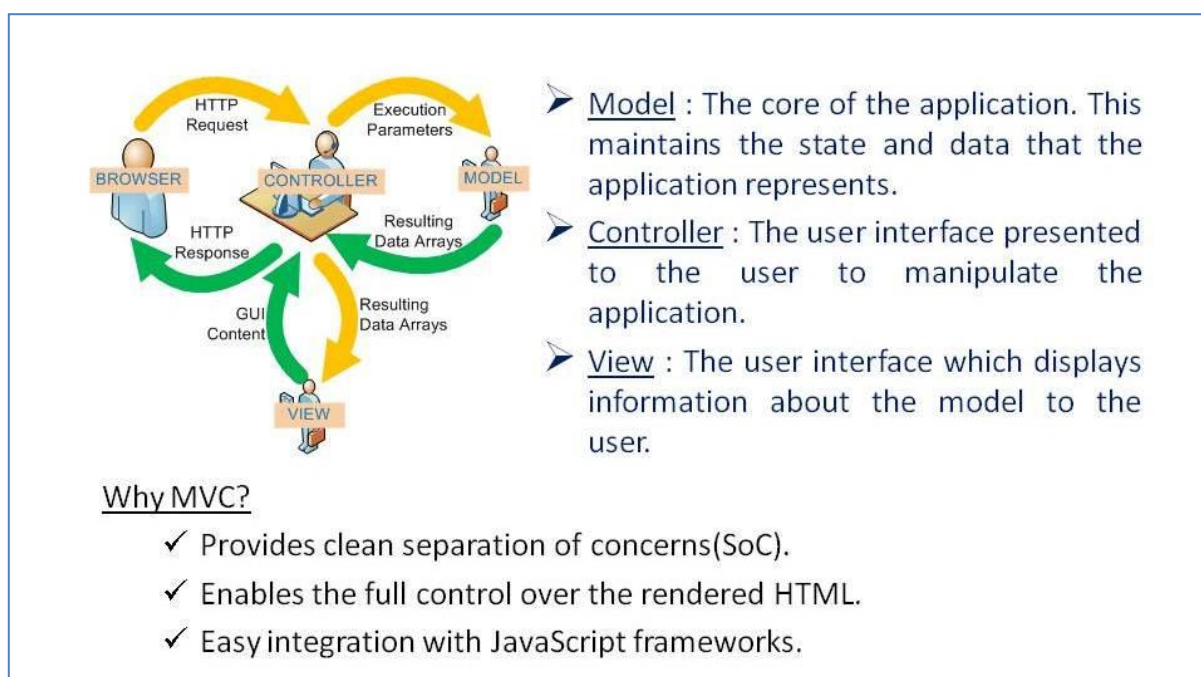


Figure 21: Model -View - Controller (MVC) paradigm.

Taking into account the aforementioned criteria, we have chosen Django application framework to act as the “heart” of our implementation. Additional software components were chosen to be used for the implementation of several features and functionalities of the system. The software components and the conferred advantages that dictated their choice are presented in the following section.

4.7 Software Architecture

The model/tool repository and the *in silico* trial repository are web based applications that have several components each one having different responsibilities inside the system.

Data Layer: We have chosen MySQL for storing the structural data of the repositories. For maintenance reasons we have chosen to store the files in a separate, independent of the RDMS, file based repository.

Business logic and data access layer: As mentioned above we have chosen Django application framework to act as the “heart” of our implementation. The advantages of Django that led to our decision are presented in the following section.

Presentation Layer: The presentation layer consists of two interfaces, the GUI and the web services. To enhance the user experience we have used two feature-rich javascript frameworks, Bootstrap and jQuery. For the REST-ful web services we have used Tastypie.



Figure 22: Software architecture of model/tool repository and of *in silico* trial repository.

4.8 Software components

MySQL [20]

MySQL is the world's most popular open source database.



Key features of MySQL are:

- **Scalability and Flexibility.** The MySQL database server provides the ultimate in scalability, sporting the capacity to handle deeply embedded applications with a footprint of only 1MB to running massive data warehouses holding terabytes of information. Platform flexibility is a stalwart feature of MySQL with all flavors of Linux, UNIX, and Windows being supported. And, of course, the open source nature of MySQL allows complete customization for those wanting to add unique requirements to the database server.
- **High Performance.** With high-speed load utilities, distinctive memory caches, full text indexes, and other performance-enhancing mechanisms, MySQL offers all the right ammunition for today's critical business systems.
- **High Availability.** MySQL offers a variety of high-availability options from high-speed master/slave replication configurations, to specialized Cluster servers offering instant failover, to third party vendors offering unique high-availability solutions for the MySQL database server.
- **Robust Transactional Support.** MySQL offers one of the most powerful transactional database engines. Features include complete ACID (atomic, consistent, isolated, durable) transaction support, unlimited row-level locking, distributed transaction capability, and multi-version transaction support where readers never block writers and vice-versa. Full data integrity is also assured through server-enforced referential integrity, specialized transaction isolation levels, and instant deadlock detection.
- **Web and Data Warehouse Strengths.** Features like main memory tables, B-tree and hash indexes, and compressed archive tables that reduce storage requirements by up to eighty-percent make MySQL a strong standout for both web and business intelligence applications.

- **Strong Data Protection.** In terms of database authentication, MySQL provides powerful mechanisms for ensuring only authorized users have entry to the database server. SSH and SSL support are also provided to ensure safe and secure connections. Backup and recovery utilities provided through MySQL and third party software vendors allow for complete logical and physical backup as well as full and point-in-time recovery.
- **Comprehensive Application Development.** Within the database, support can be found for stored procedures, triggers, functions, views, cursors, ANSI-standard SQL, and more. For embedded applications, plug-in libraries are available to embed MySQL database support into nearly any application. MySQL also provides connectors and drivers (ODBC, JDBC, etc.) that allow all forms of applications to make use of MySQL as a preferred data management server.

Django [21]



Django is a free and open source web application framework, written in Python, which follows the model–view–controller (MVC) architectural pattern. It encourages rapid development and clean, pragmatic design. It allows high-performing, elegant Web application building.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes reusability and “pluggability” of components, rapid development, and the principle of don't repeat yourself (DRY). Python is used throughout, even for settings, files, and data models.

The core Django framework consists of:

- An object-relational mapper which mediates between data models (defined as Python classes) and a relational database (“Model”).
- A system for processing requests with a web templating system (“View”).
- A regular-expression-based URL dispatcher (“Controller”).
- An optional automatic administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.

- A lightweight, standalone web server for development and testing.
- A form serialization and validation system which can translate between HTML forms and values suitable for storage in the database.
- A template system that utilizes the concept of inheritance borrowed from object-oriented programming
- A caching framework which can use any of several cache methods.
- Support for middleware classes which can intervene at various stages of request processing and carry out custom functions.
- An internal dispatcher system which allows components of an application to communicate events to each other via pre-defined signals.
- An internationalization system, including translations of Django's own components into a variety of languages.
- A serialization system which can produce and read XML and/or JSON representations of Django model instances.
- A system for extending the capabilities of the template engine.
- An interface to Python's built in unit test framework.

Tastypie [24]

Tastypie is a webservice API framework for Django. It provides a convenient, yet powerful and highly customizable abstraction for creating REST-style interfaces. Tastypie makes exposing models easy, but gives full control over what it is exposed, letting abstract away the database as much as needed. Tastypie also makes it easy to integrate with non-ORM data sources. Tastypie is BSD licensed & plays nicely with third-party apps without needing to modify their sources.

Features:

- Full GET/POST/PUT/DELETE/PATCH support.

- Reasonable defaults.
- Designed to be extended at every turn.
- Includes a variety of serialization formats (JSON/XML/YAML/bplist).
- Hypermedia as the Engine of Application State (HATEOAS) by default.
- Well-tested & well-documented.

4.9 Data flow and integration

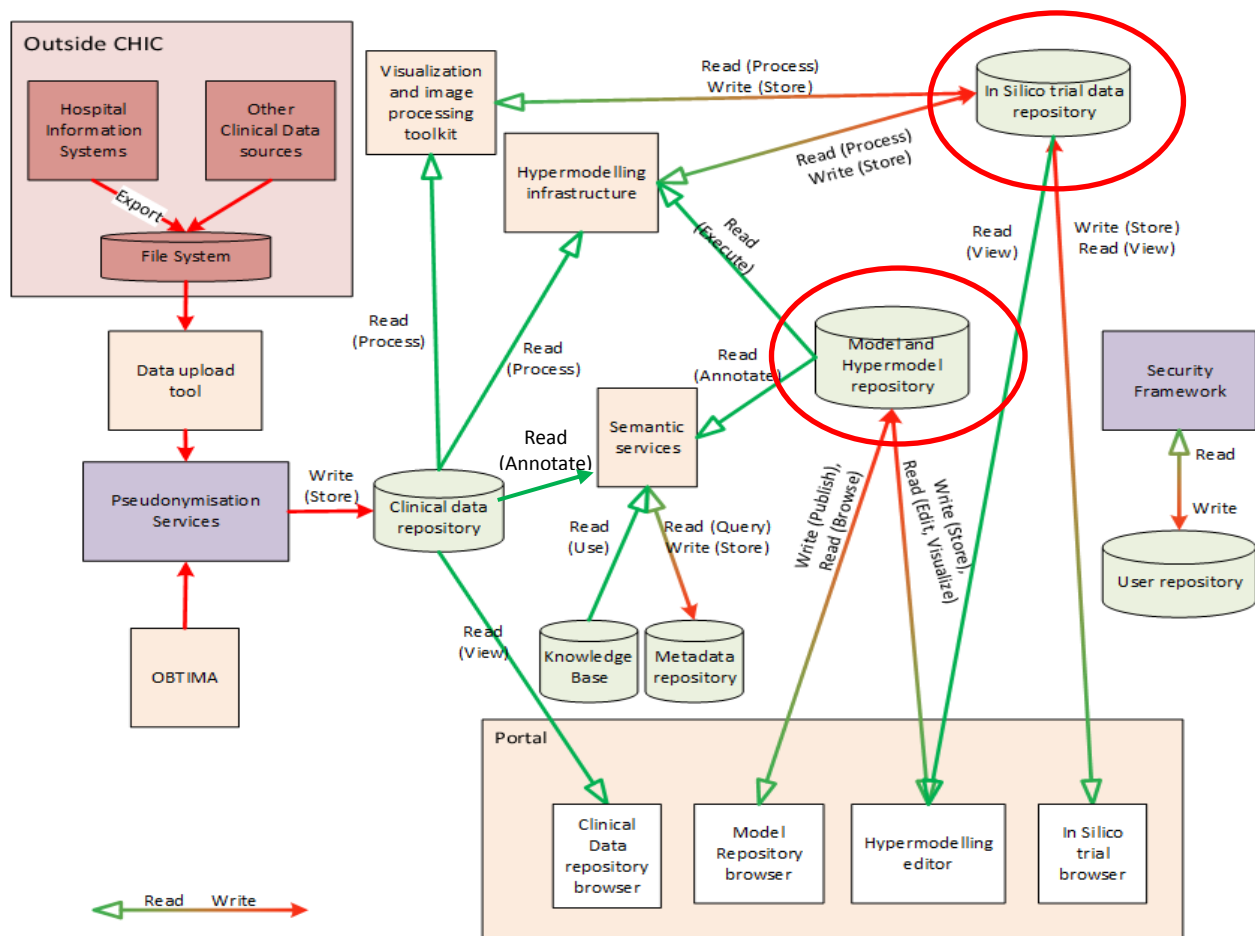


Figure 23: Data flow and interaction between CHIC components.

5 RDF storage solution for semantic metadata

5.1 Introduction

Semantic metadata are machine processable descriptions of models. In the CHIC environment, metadata is produced by representing models and annotating them, and their parts, to record their interpretation. The scope of the annotations is defined through (i) referencing of existing ontologies already supported by the community, as well as (ii) the consolidation into an ontology of the bootstrapping folksonomy of models in CHIC. The required ontological resources will support the explicit, unambiguous annotation of model characteristics as part of the model's logic (e.g. formal type of model and parts) as well as biological interpretations of model variables, parameters and data (e.g. biological quantities tied to anatomical locations).

The baseline solution in CHIC builds upon the results of the VPH RICORDO project [13] that focused on the development of RDF metadata storage and ontology-based querying of metadata for models and data in the broad range of VPH domains. Illustrations of the intended CHIC design are drawn from the documentation of this effort. The RICORDO solution has been adopted and pursued in a number of projects, including the IMI's DDMoRe [14] from which additional illustrations are borrowed.

5.2 Recording format for semantic metadata

Semantic metadata is part of a knowledge representation and numerous formats are available. In CHIC, records of semantic metadata will be maintained in the RDF format. RDF is the Resource Description Framework [15], a W3C recommendation that is pervasively used on the Web and the backbone of the Semantic Web [16]. CHIC benefits from the large RDF ecosystem in support of semantic integration and interoperability.

5.3 *RDF Store*

A metadata store can be seen as an RDF database, sometimes called a 'triple store', for which numerous systems exist. Triple stores support the exposure of RDF records in queryable databases over the Web.

In CHIC, the querying of ground annotations of models is expanded with adjunct inferencing over a knowledge base containing the ontologies used in the annotation of models. The semantic storage of model annotation is however limited to the storage of ground model descriptions.

A RDF store ought to be deployed on a dedicated server or using a virtual hosting solution for the deposition of CHIC metadata. The degree of access to the RDF store may be more permissive than the stores holding clinical data, models or other sensitive resources.

5.4 *Interface to the store*

Access to annotations about models contained in the semantic store is established by so-called SPARQL endpoints. SPARQL [17] is the query language for RDF and systems implementing it provide a query functionality that allows for inferencing, although this inferencing may be somewhat limited compared to ontology based reasoning.

RDF-Store WS – Template based querying

Example template getResourcesByType:

```
define input:inference "ricordo_rule"
SELECT ?s
FROM <http://ricordo.com>
WHERE { ?s <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <[]> }
```

Input:

http://www.ebi.ac.uk/ricordo/toolbox/sbmlo#SBMLModel

SPAQL query:

```
define input:inference "ricordo_rule"
SELECT ?s
FROM <http://ricordo.com>
WHERE { ?s <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://
www.ebi.ac.uk/ricordo/toolbox/sbmlo#SBMLModel> }
```

Figure 24: Example of SPARQL query in the RICORDO implementation.

It is also possible to produce more generic REST services allowing query to the database.

getResourcesByType

Using curl:

```
curl -i -X POST -HContent-type:application/xml --data "<query><query>http://www.ebi.ac.
```

Using RESTClient

```
Method: POST
URL: http://www.ebi.ac.uk/ricordo-rdfstore-ws/service/search/getResourcesByType
Headers: Content-Type application/xml
Body: <query><query>http://www.ebi.ac.uk/ricordo/toolbox/sbmlo#SBMLModel</query></query>
```

```
?s
http://www.ebi.ac.uk/ricordo/toolbox/sbmlo#BIOMD0000000001
http://www.ebi.ac.uk/ricordo/toolbox/sbmlo#BIOMD0000000002
http://www.ebi.ac.uk/ricordo/toolbox/sbmlo#BIOMD0000000003
http://www.ebi.ac.uk/ricordo/toolbox/sbmlo#BIOMD0000000004
http://www.ebi.ac.uk/ricordo/toolbox/sbmlo#BIOMD0000000005
http://www.ebi.ac.uk/ricordo/toolbox/sbmlo#BIOMD0000000006
http://www.ebi.ac.uk/ricordo/toolbox/sbmlo#BIOMD0000000007
```

Figure 25: Example of REST invocation in the RICORDO implementation.

5.5 Semantic metadata in context (for information only)

Semantic metadata is produced through the annotation of models. We can say that there are two kinds of metadata at production time. There is metadata that may be automatically produced through the scripted or inferential processing of model representations. There is also the metadata that is not readily elicited by the processing of the formal structure of a model and is more properly interpretative. For that second kind of metadata, expert (modeller) annotations are required. In support of the annotation of models by experts, tools have to be provided which enable the modeller to indicate the characteristics of models, their parts and data. Such tools allow the mapping of models, on the one hand, onto ontology terms on the other, such that these terms provide unambiguous, reference interpretations.

The encoded result is then deposited in the metadata store.

The metadata store will be provided with APIs that may be used by two kinds of tools: i) tools for semantic metadata management (including querying and editing or validating of metadata) and ii) other tools that require access to the information contained in the metadata store. Tools of the first kind will be Graphical User Interfaces for the management of semantic metadata and are illustrated below. The same APIs will be reusable by tools of the second kind that are developed in the project and for which information flow is needed from the semantic metadata store.

Index	Model URL	Frequency
1	http://www.ebi.ac.uk/ricordo/toolbox/sbmlo#BIOMD0000000081	1
2	http://www.ebi.ac.uk/ricordo/toolbox/sbmlo#BIOMD0000000101	1
3	http://www.ebi.ac.uk/ricordo/toolbox/sbmlo#BIOMD0000000161	1
4	http://www.ebi.ac.uk/ricordo/toolbox/sbmlo#BIOMD0000000162	1
5	http://www.ebi.ac.uk/ricordo/toolbox/sbmlo#BIOMD0000000182	1
6	http://www.ebi.ac.uk/ricordo/toolbox/sbmlo#BIOMD0000000255	1

[Tutorial](#) [RICORDO](#) [Contact Us](#)

Figure 26: Example of Web-based interface (querying) from RICORDO implementation.

Figure 27: Example of desktop interface (annotation) from DDMoRe implementation.

Figure 28: Example of Web-based interface (annotation).

5.6 Semantic metadata lifecycle in CHIC

1. The first stage of the semantic metadata lifecycle in the CHIC system is the generation of annotations:

- I. A range of annotations may be generated automatically by processing models to produce an RDF representation.

- II. A range of annotations may be generated manually using Graphical User Interface tools to produce an RDF representation.
2. The second stage of the lifecycle of semantic metadata in the CHIC system is the population and update of the repository of semantic metadata:
 - I. Generated annotations may be saved to RDF files and these files are then uploaded to the semantic metadata repository to be part of the metadata database.
 - II. Generated annotations may also be sent directly to the RDF repository from a tool used to generate them.
3. The third stage in the lifecycle of the semantic metadata in the CHIC system is the querying of the semantic metadata repository:
 - I. Querying of the semantic metadata repository may be performed programmatically, sending direct requests to the repository's SPARQL endpoint or using APIs connecting to the repository.
 - II. Querying of the semantic repository may be performed by a user of a tool which presents a graphical front-end to the semantic repository, implementing APIs connecting to the repository.

The above stages form the basis of metadata management in the context of a deployed semantic metadata repository. Additional tasks may include validation and verification of automatic or user-generated annotations.

6 Conclusion

The design of the clinical data repository has been based on the concept of a generic object to easily accommodate different types of data. In its initial implementation the system will be able to store medical images, clinical study data, histologic data and genetic data. However, due to this flexible design, the adaptation of the system for new data types can be easily done. On the other hand, a specific import system is planned for each type of data, in order to extract the relevant information from the uploaded file and to allow its annotation with the appropriate metadata.

The development of the prototype system has been initiated and integrates the EU-CHIC security framework. It is essential to take into account the legal and security aspects from the beginning of the development to provide the users with access to the data as quickly as possible. This environment also allows taking into account the ethical and legal requirements for sharing the data as described in deliverable “D4.1: Initial analysis of the ethical and legal requirements for sharing of data” and “D4.3.1: Development of the data protection and copyright framework for CHIC - first iteration”.

The model/tool repository is the technological platform where hypomodels, hypermodels, data transformation tools, linkers and other tools used in the construction and execution of hypermodels are stored. End users and other technological components may access the stored models/tools, or store new ones, using the graphical or programmatic interfaces provided by the model/tool repository.

The *in silico* trial repository is the technological platform where the data related to *in silico* trials and *in silico* experiments are stored. The design of the *in silico* trial repository is based on the classical and standardized principles of the real clinical trials. Similar to the model/tool repository, end users and other technological components may access the *in silico* trial repository in order to prepare new *in silico* trials or to store the results of the conducted *in silico* experiments. The crucial role of repeatability and reproducibility has been explained in this report.

The model/tool repository and the *in silico* trial repository are web based applications that have several components each one having different responsibilities inside the system. In this report we have presented the Entity-Relationship (ER) diagrams of each of the aforementioned repositories, along with the concepts and the software components used in the implementation of the backend and the frontend of these repositories.

Last but not least, in this deliverable we indicated the design for the semantic metadata repository as part of the CHIC system. This repository takes the form of an RDF database supporting basic metadata management functionalities. APIs to the repository will be reusable by other tools and in support of metadata information flow within the CHIC system. The metadata repository is an element of the metadata CHIC framework. The framework will also include a knowledge base of ontologies supporting reasoning and query expansion of the metadata repository. Earlier into the lifecycle of the metadata, the framework will also include supporting metadata management tools geared toward expert modelling users.

The repositories will be deployed on the cloud infrastructure developed in WP5 and described in deliverable *“D5.1.1: The CHIC technical architecture – initial version”*. To this aim, virtual environments will be used to run the different services required for the repositories. The first services that will be available to the users will concern the medical images followed by the clinical study data. A priority will also be set on the REST interface, which will be used by the pseudonimization and data upload tool developed by CUSTODIX as part of WP5. This step is critical to populate the database with relevant data collected by the clinical partners.

The graphical and programmatic interfaces of the four aforementioned repositories will be reported in *“D8.2: Prototype implementation of the CHIC repositories”*, *“D8.3: Implementation of the interfaces of the CHIC repositories”* and *“D10.2: Design of the orchestration platform, related components and interfaces”*.

7 References

- [1] DICOM file format standard. <http://medical.nema.org/standard.html>
- [2] Metalmage file format. <http://www.itk.org/Wiki/ITK/MetalO/Documentation>
- [3] Mayo Clinic. “ANALYZE 7.5 File Format”. September 27, 2007.
- [4] Niftii file format. <http://nifti.nimh.nih.gov/nifti-1>
- [5] CDISC ODM standard. <http://www.cdisc.org/odm>
- [6] Toby C. Cornish. “An introduction to digital whole slide imaging and whole slide image analysis”. <http://www.hopkinsmedicine.org>
- [7] Kistler M, Bonaretti S, Pfahrer M, Niklaus R, Büchler P. “The Virtual Skeleton Database: An Open Access Repository for Biomedical Research and Collaboration”.
<http://www.jmir.org/2013/11/e245>
- [8] FMA ontology. <http://fma.biostr.washington.edu>
- [9] Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides. “Design Patterns: Elements of Reusable Object-Oriented Software”. October 21, 1994.
- [10] Repository pattern. <http://msdn.microsoft.com/en-us/library/ff649690.aspx>
- [11] Unit of Work pattern. <http://martinfowler.com/eaCatalog/unitOfWork.html>
- [12] Role-based security. [http://msdn.microsoft.com/en-us/library/shz8h065\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/shz8h065(v=vs.110).aspx)
- [13] <http://www.ricordo.eu>
- [14] <http://www.ddmore.eu>
- [15] <http://www.w3.org/RDF>
- [16] <http://www.w3.org/2001/sw>
- [17] <http://www.w3.org/TR/rdf-sparql-query>
- [18] <http://p-medicine.eu/tools/obtima>
- [19] <https://www.openclinica.com>
- [20] <https://www.mysql.com>
- [21] <https://www.djangoproject.com>
- [22] http://en.wikipedia.org/wiki/Django_%28web_framework%29
- [23] <http://en.wikipedia.org/wiki/HATEOAS>
- [24] <http://tastypieapi.org>
- [25] Patterns of Enterprise Application Architecture. With David Rice, Matthew Foemmel, Edward Hieatt, Robert Mee, and Randy Stafford. Addison-Wesley. ISBN 0-321-12742-0.
- [26] McNutt, M. (2014). Reproducibility. Science, 343, 229. doi:10.1126/science.1250475

Appendix – Abbreviations and acronyms

<i>ACID</i>	Atomic, Consistent, Isolated, Durable
<i>ADO</i>	ActiveX Data Objects
<i>ANSI</i>	American National Standards Institute
<i>API</i>	Application Programming Interface
<i>BSD</i>	Berkeley Software Distribution
<i>CDISC</i>	Clinical Data Interchange Standards Consortium
<i>CHIC</i>	Computational Horizons in Cancer
<i>CSS</i>	Cascading Style Sheets
<i>CT</i>	Computed Tomography
<i>GUI</i>	Graphical User Interface
<i>DDMoRe</i>	Drug Disease Model Resources
<i>DICOM</i>	Digital Imaging and Communications in Medicine
<i>DOI</i>	Digital Object Identifier
<i>ER</i>	Entity-Relationship
<i>FGED</i>	Functional Genomics Data Society
<i>FMA</i>	Foundational Model of Anatomy
<i>HATEOAS</i>	Hypermedia as the Engine of Application
<i>HTML</i>	HyperText Markup Language
<i>HTTP</i>	Hypertext Transfer Protocol
<i>IdP</i>	Identity Providers
<i>IIS</i>	Internet Information Service
<i>IMI</i>	Innovative Medicines Initiative
<i>ISO</i>	International Organization for Standardization
<i>ITK</i>	Insight Segmentation and Registration Toolkit
<i>JDBC</i>	Java Database Connectivity
<i>JSON</i>	JavaScript Object Notation
<i>JPEG</i>	Joint Photographic Experts Group
<i>MAGE-ML</i>	Microarray Gene Expression - Markup Language
<i>MAGE-TAB</i>	Microarray Gene Expression - Tabular Format

<i>MIAME</i>	Minimum Information About a Microarray Experiment
<i>MINSEQE</i>	Minimum Information about a high-throughput SEQuencing Experiment
<i>MRI</i>	Magnetic Resonance Imaging
<i>MS SQL</i>	Microsoft Structured Query Language
<i>MVC</i>	Model-View-Controller
<i>ODBC</i>	Open Database Connectivity
<i>ODM</i>	Operational Data Model
<i>OID</i>	Object Identifier
<i>ORM</i>	Object-Relational Mapping
<i>PACS</i>	Picture Archiving and Communication System
<i>PMID</i>	PubMed identifier
<i>RDF</i>	Resource Description Framework
<i>REST</i>	Representational State Transfer
<i>SP</i>	Service Provider
<i>SPARQL</i>	Simple Protocol and RDF Query Language
<i>SSH</i>	Secure Shell
<i>SSL</i>	Secure Sockets Layer
<i>SSO</i>	Single Sign-On
<i>TIFF</i>	Tagged Image File Format
<i>UoW</i>	Unit of Work
<i>US</i>	Ultrasound Scan
<i>VPH</i>	Virtual Physiological Human
<i>WML</i>	Wireless Markup Language
<i>WP</i>	Work Package
<i>WSI</i>	Whole Slide Imaging
<i>W3C</i>	World Wide Web Consortium
<i>XHTML</i>	Extensible HyperText Markup Language
<i>XML</i>	Extensible Markup Language