CHIC

Computational Horizons in Cancer

Deliverable No. 10.2

Design of the orchestration platform, related components and interfaces

Grant Agreement No.:	600841
Deliverable No.:	D10.2
Deliverable Name:	Design of the orchestration platform, related components and interfaces
Contractual Submission Date:	30/09/2014
Actual Submission Date:	04/12/2014

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

COVER AND CONTROL PAGE OF DOCUMENT	
Project Acronym:	**CHIC**
Project Full Name:	Computational Horizons In Cancer (CHIC): Developing Meta- and Hyper-Multiscale Models and Repositories for In Silico Oncology
Deliverable No.:	D10.2
Document name:	Design of the orchestration platform, related components and interfaces
Nature (R, P, D, O)[1]	R
Dissemination Level (PU, PP, RE, CO)[2]	PU
Version:	1.0
Actual Submission Date:	04/12/2014
Editor: Institution: E-Mail:	Georgios Aravanis Philips georgios.aravanis@philips.com

ABSTRACT:

This deliverable aims to present the design and the architecture of some of the critical components of the CHIC hypermodelling infrastructure. In particular we define the requirements and the initial design of the CHIC hypermodelling editor, which is the end user software component for designing, triggering the execution, and retrieving the results of the CHIC integrative hypermodels. Furthermore, we describe the programmatic interfaces for accessing the CHIC repositories and retrieving information about the models, hypermodels, and data for the purpose of designing new hypermodels. To demonstrate the integration of the different components of the hypermodelling infrastructure we present two use cases, the first focusing on the design of a new hypermodel while the second concentrates on the execution of a ready hypermodel.

KEYWORD LIST:

CHIC orchestration, data, requirements, user Interfaces, programmatic Interfaces, repositories

[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 no 600841.

MODIFICATION CONTROL

Version	Date	Status	Author
0.1	12/09/2014	Draft	Georgios Aravanis
0.2	25/09/2014	Draft	Georgios Aravanis
0.3	24/11/2014	Draft	Georgios Aravanis
0.4	25/11/2014	Draft	Stelios Sfakianakis
0.5	28/11/2014	Draft	Georgios Aravanis
0.9	01/12/2014	Pre-final	Stelios Sfakianakis
1.0	04/12/2014	Final	Georgios Aravanis

List of contributors

- Georgios Aravanis, PHILIPS
- Stelios Sfakianakis, FORTH
- Fay Misichroni, ICCS

Table of Contents

List of Figures

1 Executive Summary

The purpose of this document is to report on the design and the architecture of some of the critical components of the CHIC hypermodelling infrastructure. In particular we define the requirements and the initial design of the CHIC hypermodelling editor, which is the end user software component for designing, triggering and monitoring the execution, and retrieving the results of the CHIC integrative hypermodels. The editor, however, is not an autonomous component in the CHIC platform: it builds upon and integrates with the data, model, and hypermodel repositories and the CHIC hypermodel execution framework. In the present document we therefore also describe the programmatic interfaces for accessing the CHIC repositories for the purpose of designing new hypermodels. These interfaces are compatible with the underlying design of these repositories and are presented in a technology neutral way, so that different middleware technologies (e.g. Web Services or other) can be used for their realization.

We conclude this document with two scenarios that exhibit the integration of the different components and the use of the programmatic interfaces. The first showcases the design of a new hypermodel by fetching the descriptions of the component models from the corresponding repositories. The second use case is the execution of a ready hypermodel, which requires a close cooperation between the hypermodelling editor, the model and inSilico Trial repositories, and the execution framework.

2 Introduction

The CHIC project aims to build a technological platform for advanced VPH research, focusing on the construction, execution, and reuse of VPH mathematical and computational hypermodels. The envisaged hypermodelling infrastructure consists primarily of the execution framework, the model and data repositories, and the hypermodelling editor (Figure 1).

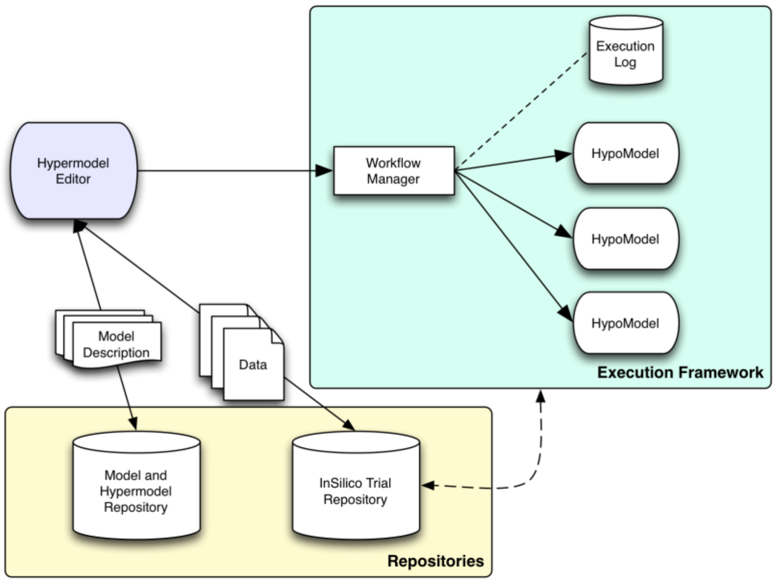


Figure 1 A simplified view of the CHIC orchestrated platform

The responsibilities of the major subsystems that comprise the CHIC hypermodelling infrastructure are the following:

- The Execution Framework provides the engine for the actual execution of the hypermodels in the CHIC computational cloud. This is where the hypomodels are run, exchanging data and signals, and producing the final results of the hypermodels.

- The model and data repositories are the core components of the CHIC data and model management layer. They provide query-based retrieval of the model and data descriptions in compliance with the metadata schemas adopted in CHIC.

- The Hypermodelling editor is the front-end to this infrastructure. The editor allows the end users (computational biologists, researchers, etc.) to design new hypermodels and then submit them to the execution framework in order to test their research hypotheses or validate their assumptions.

In this document we mostly focus on the design and general management of the hypermodels and the technologies to support these use cases. Details about the execution framework and the model repositories can be found in the relevant work packages (Work package 7 and 8 correspondingly). Therefore we concentrate on the initial requirements for the implementation of the hypermodelling editor (Section 3) and the necessary “glue” for retrieving and storing models and hypermodels in the form of the “programmatic interfaces” (Sections 4 and 5). Finally, in Section 6 we provide some exemplary uses cases for the use of these interfaces.

3 Data and hypermodel Orchestration

The CHIC Hypermodelling editor is the end user application for designing the CHIC hypermodels that can be subsequently run in the CHIC execution infrastructure. In the following sections we discuss the requirements that led to the design of the first prototype of the Hypermodelling Editor, its position in the CHIC architecture and the components it interacts with, and the implementation of the first prototype.

3.1 Initial requirement analysis

The CHIC hypermodelling editor is the primary software component in the CHIC platform used by computational biologists, system biology scientists, and other domain experts for designing and triggering the execution of the hypermodels. According to Deliverable 7.101,

> *"A hypermodel (or composite model) is the composition and orchestration of multiple hypomodels (single models either scientific or computer ones). […] The hypomodels capture the existing knowledge about a portion of the process, typically at a characteristic space-time scale. The hypermodel can simulate an entity or phenomenon that may be more complex than the ones simulated by each separate simpler hypomodel."*

The defining property of a hypermodel is that is composed of hypomodels, and on the other hand a hypomodel is a model that is used for the composition of hypermodels. Therefore these definitions are relative: a hypermodel can also be a hypomodel if it is used as a component to the construction of another hypermodel. Effectively, the containment relationship between hypomodels and hypermodels can be used to build a hierarchy of nested hypermodels, similar to the "*matryoshka dolls*" (Russian nesting dolls).

Another perspective for describing models relates to their realization as computational entities. In the CHIC platform the models are described in two main layers (Figure 2):

1. As concrete computational entities that are "actionable" (can be invoked and executed), and
2. As "Metamodels" in the "abstract" world where their domain functionality, semantics, etc. are defined

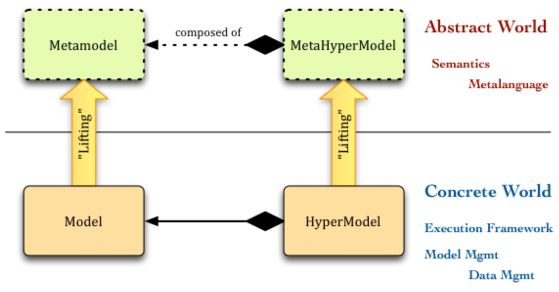


Figure 2 The different model description and composition aspects

In the "computational" or "concrete" world shown in the bottom of the figure above, the models are defined based on their computational aspects, or example execution properties and requirements such as the programming language or environment used, operating system, etc. These aspects are important for the CHIC Execution Framework provided by WP7 but not so much for the Hypermodelling Editor. In the hypermodelling editor the users should not be aware of the execution details of each hypomodel: the underlying implementation complexity is irrelevant for the design of hypermodels. Therefore, the Editor mostly "deals" with the upper level where the models are described based on their semantics and their domain specific characteristics (such as their temporal or spatial scales[3]). Nevertheless, the editor should also be aware of some of the implementation details of each hypomodel in order to make sure that the CHIC execution framework can execute the designed hyper models.

Even considering the "abstract" metamodels, when two hypomodels are put together there is the issue on how they interact with each other in order to achieve the objective of the hypermodel that they compose. For this, we have adopted a paradigm originated in computer programming: Each model is represented as a "callable unit" that accepts a set of input values ("parameters" or "arguments") and returns a set of output values. This approach is actually the "generic stub" concept of the CHIC Hypermodelling Framework[4] that aims at exposing each model in a standardized way, by providing a unified interface for the model execution. A graphical representation of this concept is shown in Figure 3.

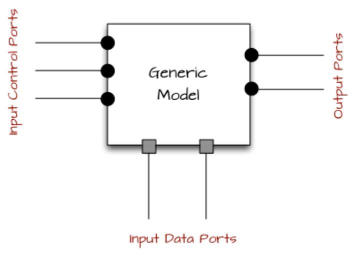


Figure 3 The generic interface of a computational model

The different "ports" shown in the figure correspond to communication channels through which data and control flows between the model and its environment. In essence these ports are "*affordances*"[5] that define how the model can possibly be used when composed with other hypomodels, because they allow for the interconnection of models. *The way composition of hypermodels is supported is therefore through the connection of the output ports of one hypomodel to the input ports of another*. Consequently, these connection links provide exchange of data between the hypomodels and as an added value some basic coordination of the models can be performed since a subsequent model cannot run unless its predecessor models have provided values for all of its inputs.

The data dependencies among the models are the first coordination strategy we implement. It is certainly not the only one possible. In more complex hypermodels the execution of a hypomodel may depend on a condition (e.g. another hypomodel producing a value above some threshold) or the hypermodel can contain cycles where a series of hypomodels need to be run repeatedly until some

[3] Deliverable 6.1 "Cancer hypomodelling and hypermodelling strategies and initial component models"

[4] Deliverable 7.1 "Hypermodelling specifications"

[5] This is a term introduced by psychologist James J. Gibson originally in his 1977 article "The Theory of Affordances", where he says: "*The affordances of the environment are what it offers ... what it provides or furnishes, either for good or ill.*"

condition becomes true. These are additional requirements that need to be addressed as soon as such exemplary hypermodels are defined in CHIC.

3.2 The Editor in the CHIC architecture

The Hypermodelling Editor is an end-user application, which means that it features a user interface and it's compliant with the portal framework adopted in CHIC[6]. Being a user-facing architectural component it happily resides in the User Interface layer as shown in the figure below.

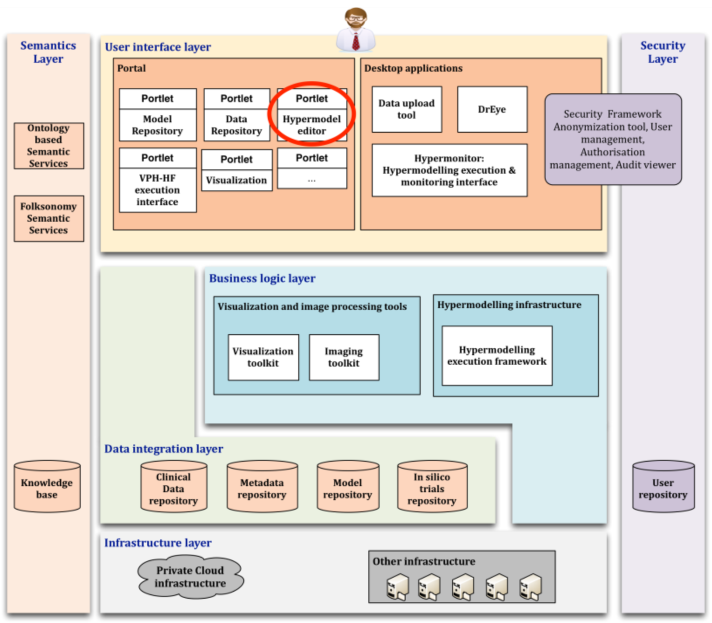


Figure 4 The Hypermodelling editor in the context of the CHIC architecture

As a component in the CHIC architecture the following are its responsibilities:

- Allows the single sign on (SSO) of the CHIC users, which means that the users that have already logged into the CHIC portal are immediately identified by the Editor and do not need to enter their credentials (passwords) again.
- Support the browsing of the available hypo- and hyper- models and their discovery using their semantics and metadata based descriptions.
- Support the composition of the hypomodels and the storage of the new hypermodels. The users are also allowed to revise the designed hypermodels and change them as they wish. This functionality, for example, enables the users to create hypermodels in multiple login sessions.
- Support the submission of hypermodels to the CHIC execution framework. The CHIC execution framework takes then the responsibility of "enacting" the hypermodels using the CHIC computational resources.

[6] Deliverable 10.1 "The CHIC portal"

- Collaborates with the execution framework to allow the users monitor the execution of their submitted hypermodels.
- Retrieve the final results of the hypermodel execution and present them to the user or direct the users in the appropriate portlet in the CHIC portal where they can be retrieved.

In order to offer its functionalities the Editor has the following "collaborators" (Figure 5):

- The CHIC model and hypermodel repository[7]. This is the where information about the models is stored and the editor is using the model repository interfaces (Section 5) to retrieve it. Additionally this is the component that offers the hypermodel persistence functionality and so the editor stores the new hypermodels therein.
- The In Silico trials repository, where information about patient specific data compatible with the models and the final results of the hypermodel execution are stored.
- The CHIC Metadata repository. This is where the semantics rich annotations of the models are stored. As described in Deliverable 8.1 this component is an RDF "triplestore" and the Editor uses the SPARQL query language to retrieve the semantic annotations of the models.
- Finally, the hypermodelling execution framework where the constructed hypermodels are sent for execution. The Editor further interacts with the Execution framework in order to get the running status updates of the submitted hypermodels.

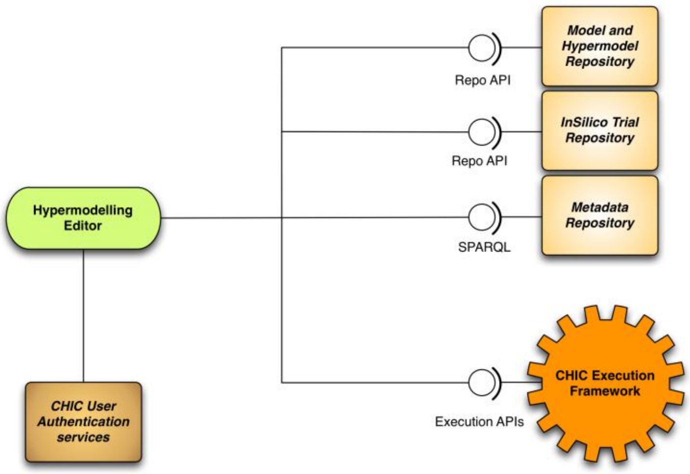


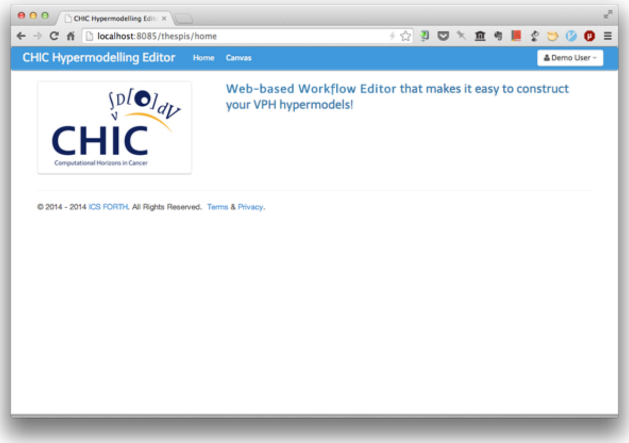
Figure 5 The components that the Hypermodelling editor collaborates with

3.3 User Interface

Designing the CHIC hypermodels is envisaged as a visual process: It means that the users have a graphical depiction of the (hypo) models that are put together and they connect them in a visual way

[7] Deliverable 8.1 "Design of the CHIC repositories"

by drawing lines using "drag n drop" facilities. The underlying paradigm is similar to the one of the visual programming environments that lets users create programs by manipulating program elements graphically rather than textually. In this sense the hypermodel design process is similar to programming where the "program elements" are the hypomodels put together to work in tandem. This diagraming programming for hypermodel building is based on the idea of "boxes and arrows" where the boxes represent models and the arrows represent communication channels between those models.



Web-based Workflow Editor that makes it easy to construct your VPH hypermodels!

© 2014 - 2014 ICS FORTH. All Rights Reserved. Terms & Privacy.

Figure 6 The initial (welcome) screen of the editor (prototype version)

The Editor features a graphical user interface built on modern web technologies (HTML5) to guarantee almost ubiquitous access. The "welcome" page of the Editor's prototype is shown in Figure 6 above. The user interacts in a visual way by "dragging and dropping" connections in the main draw area ("canvas"), as shown in the next figure.

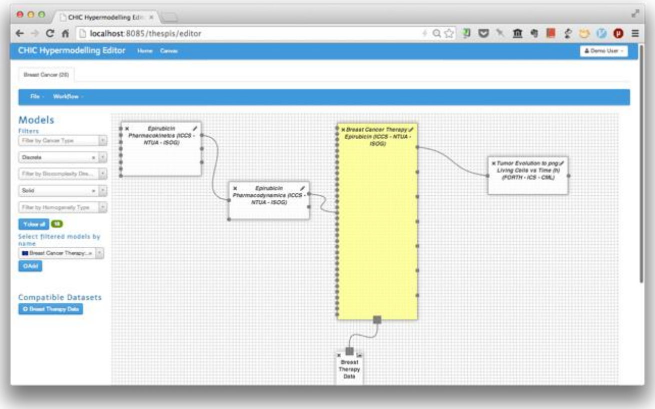

CHIC Hypermodelling Edit
localhost:8085/thespis/editor
CHIC Hypermodelling Editor
Home
Canvas
Demo User
Breast Cancer (26)
File
Workflow
Models
Filters
Filter by Cancer Type
Discrete
Filter by Biocomplexity Dim...
Solid
Filter by Homogeneity Type
Clear all
Select filtered models by name
Breast Cancer Therapy...
Add
Compatible Datasets
Breast Therapy Data
Epirubicin Pharmacokinetics (ICCS - NTUA - ISOG)
Epirubicin Pharmacodynamics (ICCS - NTUA - ISOG)
Breast Cancer Therapy Epirubicin (ICCS - NTUA - ISOG)
Tumor Evolution to png: Living Cells vs Time (h) (FORTH - ICS - CML)
Breast Therapy Data

Figure 7 The visual composition of the hypomodels into more complex hypermodels.

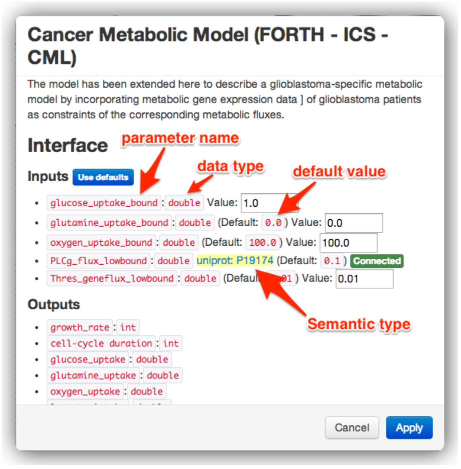
The main user activities when the users interact with the editor are the following:

- Model selection. The user is able to provide values for certain search criteria (facets), like the type of tumor or the biological level modeled (e.g. biological processes at the cell level), in order to efficiently discover the models of interest.

- Model inspection and linking. Each model added into the drawing area is represented as a box with a number of input and output ports that denote the input and output parameters and data. The user is allowed to inspect the models' details, to get more information about the syntactic and semantic type of their parameters, and their default values (Figure 8). After adding two or more models the users can link them based on the input and output parameters should their types match. The linking of two models designates exchange of information that is produced by the source model and consumed by the second one. In the case where semantics rich information about the models' parameters is available, the Editor can provide a visual "guidance" about the validity of the connection, as shown in Figure 9.

- Models execution and monitoring. When the user considers her workflow complete can save it and subsequently run it by providing any missing input values for the constituent models. At any time there are visual indications of the execution progress while, since the models are run on the CHIC computational platform, the user can logged out and reconnect in the future, without affecting the execution.

- Retrieving results of current and past executions. When the workflow ends the users are able to retrieve the results and in some cases, e.g. for plots, they can even preview those results inside their browser without downloading them. Each execution trace is stored centrally and kept for historical reasons, reproducibility, and provenance.


Cancer Metabolic Model (FORTH - ICS - CML)

The model has been extended here to describe a glioblastoma-specific metabolic model by incorporating metabolic gene expression data] of glioblastoma patients as constraints of the corresponding metabolic fluxes.

Interface

Inputs Use defaults

- glucose_uptake_bound : double Value: 1.0
- glutamine_uptake_bound : double (Default: 0.0) Value: 0.0
- oxygen_uptake_bound : double (Default: 100.0) Value: 100.0
- PLCg_flux_lowbound : double uniprot: P19174 (Default: 0.1) Connected
- Thres_geneflux_lowbound : double (Defaul 01) Value: 0.01

Outputs

- growth_rate : int
- cell-cycle duration : int
- glucose_uptake : double
- glutamine_uptake : double
- oxygen_uptake : double

Figure 8 The basic metadata descriptions for the inputs and outputs of the hypomodels

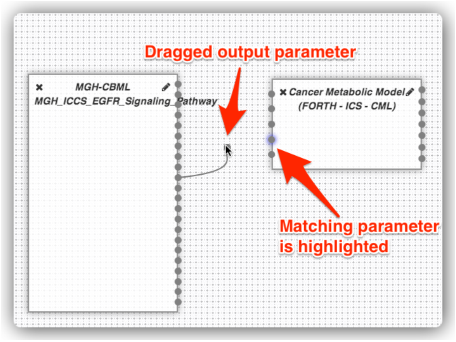

Dragged output parameter
MGH-CBML
MGH_ICCS_EGFR_Signaling_Pathway
Cancer Metabolic Model (FORTH - ICS - CML)
Matching parameter is highlighted

Figure 9 Linking two hypomodels in a "visual" way

4 Repository interoperable interfaces context

The purpose of this section is to describe the context of the designed interoperable interfaces to the model/tool and the in silico repositories for the CHIC framework[8]. The designed environment consists of a number of services and components which are already described in Section 3 and the refined interface services that will be presented in the next sections, namely:

- The ModelToolRepositoryService, responsible for communicating with the storage of models (Model/Tool Repository);

- The InSilicoTrialRepositoryService, responsible for communicating with the storage of the in silico trials (In Silico Trial Repository).

The two aforementioned repositories include all the necessary information that is needed for defining a model/hypoermodel and an in silico trial. For that reason, the interfaces to those repositories were included in the scope of *“Task 10.2: Interoperable interfaces for retrieving model and hypermodel descriptions from the corresponding repositories”*. In the clinical data and meta-data repositories, additional information is stored. The interfaces to those repositories will be reported in “*D8.3 Implementation of the interfaces of the CHIC repositories”.*

4.1 Usage context

The usage context has been described in a previous deliverable i.e. “D7.1 Hypermodelling specifications” in a higher level of detail. In this section we mention the main use cases that require the use of interfaces between different CHIC components and the underlying repositories and are presented in full detail in Section 6**.**

Three main use cases identified are the following:
- Modellers via the Hypermodelling Editor retrieve (hypo/hyper) models from the Model Repository in order to compose Hypermodels.
- Clinicians run experiments which executes Hypermodels through the Hypermodel Editor for a certain subject.
- Clinicians select and execute trials through the Hypermonitor component

The aforementioned use cases are supported by the CHIC framework and more specifically by the Hypermodelling Framework (HF) and the underlying repositories, which are accessed through the interfaces presented and analysed in the next sections of this deliverable.

[8] The design, the technologies to be used and the software architecture of the CHIC repositories are reported in *“D8.1 Design of the CHIC repositories”*

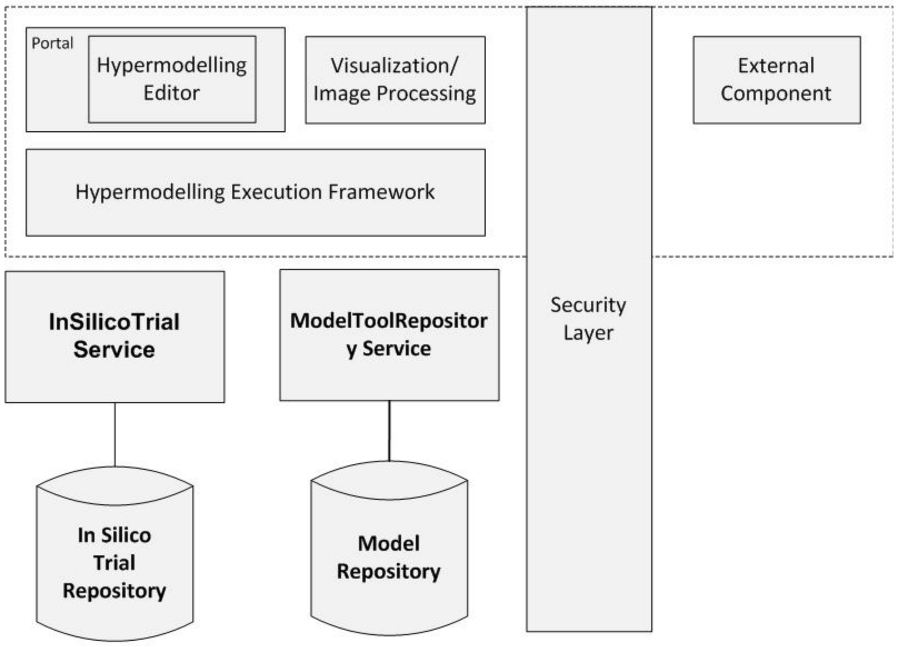

Portal
Hypermodelling Editor
Visualization/ Image Processing
External Component
Hypermodelling Execution Framework
Security Layer
InSilicoTrial Service
ModelToolRepository Service
In Silico Trial Repository
Model Repository

Figure 10 The ModelToolRepository and In SilicoTrial services in the context of the CHIC architecture

The CHIC components and any component of related research efforts, rely on an infrastructure of services (Figure 10), which are responsible for retrieving and processing data. The ModelToolRepository service is at the core of this infrastructure and provides to the framework the model parameters of a specific model. The service can receive requests from various components of the CHIC architecture. The Model/Tool Repository contains definitions of models, hypermodels, linkers, data transformation tools and model metadata – such as parameters, data types, model scripts, and semantics of model parameters.

The InSilicoTrial service provides an interface that enables the storing of in silico trials in the corresponded repository. It receives requests from the HF which includes the used hypermodel, the participating patient and the produced results which are then stored to the repository.

In order to access the services, the minimum requirements described in “D 5.2 Security guidelines and initial version of security tools” must be met. Alternatively the services should be compatible with the authentication and authorization mechanisms described in the deliverable “D8.1: Design of the CHIC repositories”.

4.2 *Provided interfaces*

The Model/Tool repository service provides an interface enabling storing/deleting/updating/retrieving a list of available models and related descriptive information, including among others the description of:

- model’s input and output parameters,
- parameters types,

- possible value ranges and measurement units of parameters,
- computational representations of model , and
- file URLs

The In Silico Trial repository service provides an interface enabling storing/deleting/updating/retrieving the data of a trial in the corresponded repository, including:

- the used hypermodel,
- the participating patient and
- the produced results of a designed experiment.

Details of the interfaces are described further in the document in the next sections 5.1, 5.2.

5 Repository interoperable interfaces services

5.1 *Model/Tool Repository Service*

This section includes the schema, an overview of the main entities and the design of the standardized interfaces of the Model/Tool Repository.

5.1.1 Model/Tool Repository Schema

The main entities of the Model/Tool Repository (Figure 11) are the tool, the parameter and the property. The tool entity includes all the descriptive information of a tool. The parameter entity contains all the information regarding the input and output parameters of a tool. The property entity contains the properties that could characterize a tool. The actual value of a property for a specific tool is stored in the tool_property entity.

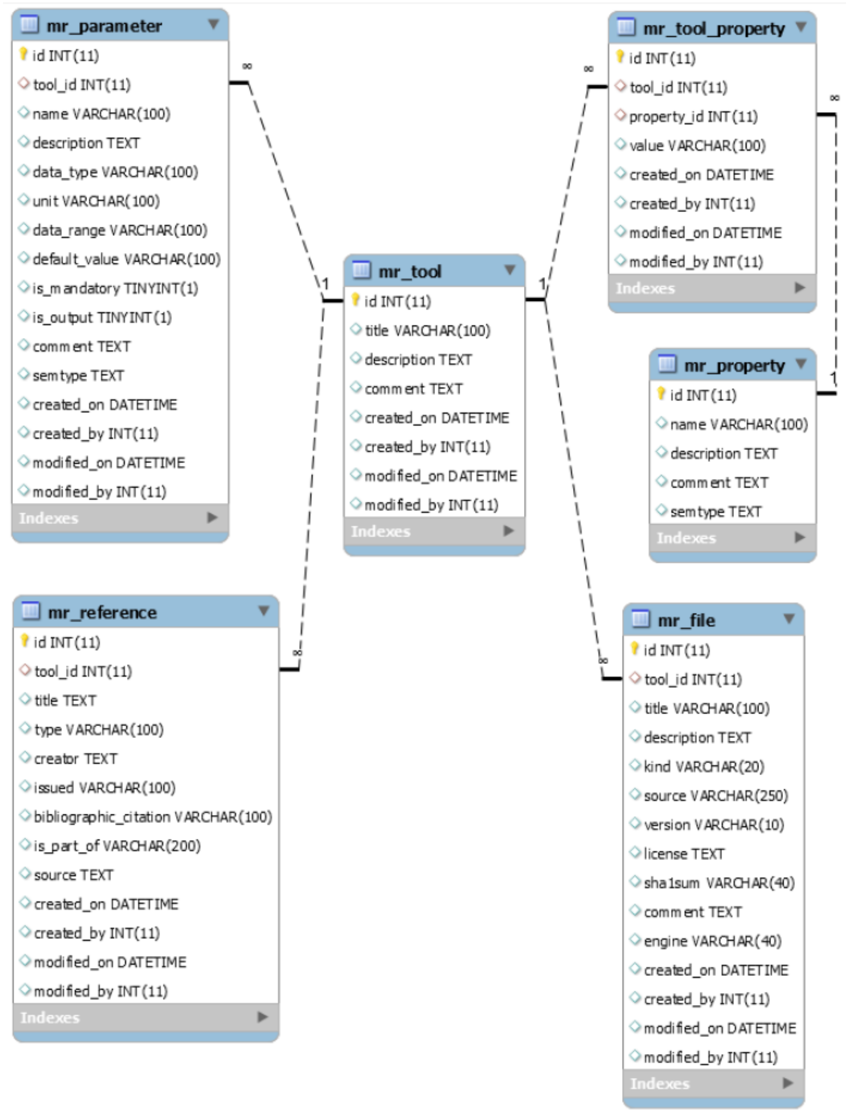


Figure 11 Entity-Relationship (ER) diagram of model/tool repository

5.1.2 Model/Tool Repository Interfaces

Tool

Method	Arguments	Returned object	Description
storeTool	• String title • String description • String comment	ToolIdResponse	This method stores the basic descriptive information of the tool and returns the id. If the id already exists, updates the values
getAllTools		AllToolsResponse	This method returns all the tools and the corresponded descriptive information stored (title, description, comment) and null when not existing
getToolById	• Int id	ToolByIdRespons e	This method returns the descriptive information stored under the id (title, description, comment) and null when not existing
deleteToolById	• Int id	ToolDeletionResul tResponse	This method deletes the descriptive information, the files, the parameters and property values of a tool. Returns Success or Fail

- **ToolIdResponse**

ToolIdResponse is of complex type IdResponse, the structure of which is presented in Figure 12.

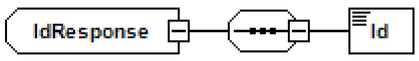


Figure 12 IdResponse Complex Type tree view

- **AllToolsResponse**

AllToolsResponse is of complex type AllToolsResponse, the structure of which is presented in Figure 13.

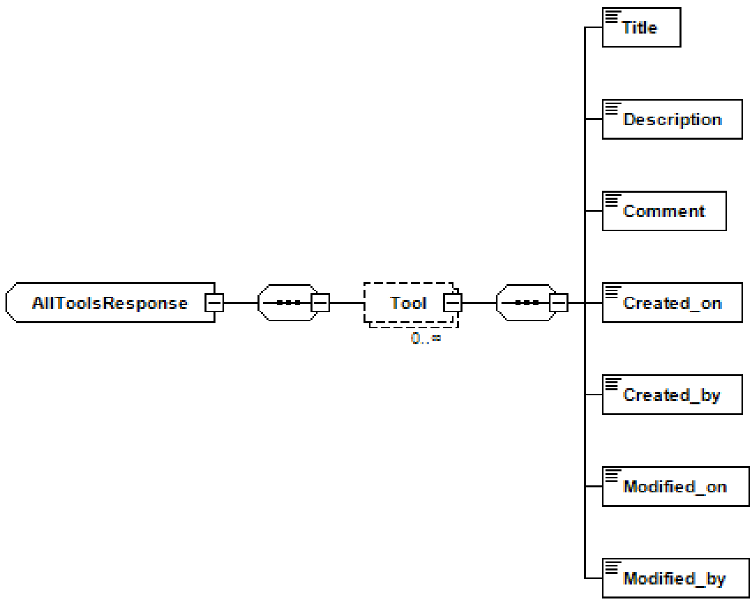


Figure 13 AllToolsResponse Complex Type tree view

- **ToolByIdResponse**

ToolByIdResponse is of complex type ToolByIdResponse, the structure of which is presented in Figure 14.

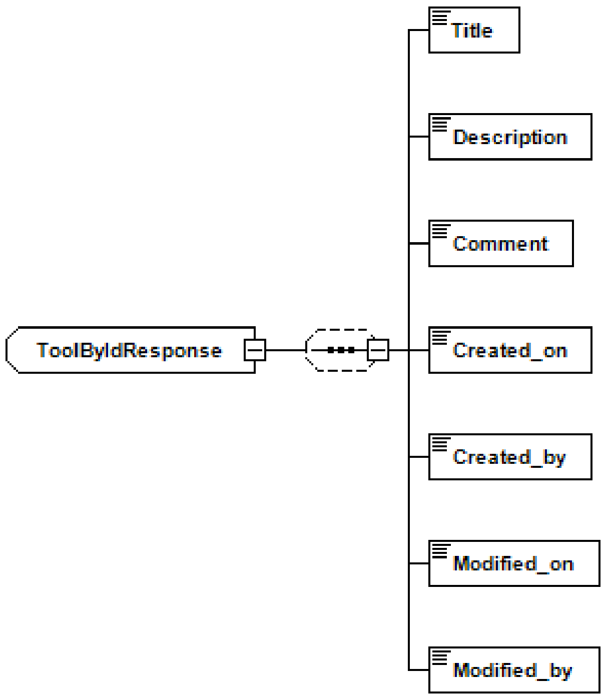


Figure 14 ToolByIdResponse Complex Type tree view

- **ToolDeletionResultResponse**

ToolDeletionResultResponse is of complex type DeletionResultResponse, the structure of which is presented in Figure 15.

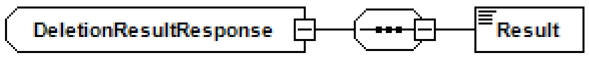


Figure 15 DeletionResultResponse Complex Type tree view

Parameter

Method	Arguments	Returned object	Description
storeParameter	• Int tool_id • String name • String description • String data_type • String unit • String data_range • String is_mandatory • String is_output • String comment	ParameterIdResponse	This method stores the parameter information of a tool and returns the id. If the id already exists, updates the values
deleteParameter	• Int id	ParameterDeletionResultResponse	This method deletes a certain parameter and returns Success or Fail
getParametersByToolId	• Int tool_id	ParametersByToolIdResponse	This method returns the information of all the parameters of a given tool
getMandatoryParametersByToolId	• Int tool_id	MandatoryParametersByToolIdResponse	This method returns the information of the mandatory parameters of a given tool
getInputParametersByToolId	• Int tool_id	InputParametersByToolIdResponse	This method returns the information of the input parameters of a given tool
getOutputParametersByToolId	• Int tool_id	OutputParametersByToolIdResponse	This method returns the information of the output parameters of a given tool

- **ParameterIdResponse**

ParameterIdResponse is of complex type IdResponse, the structure of which is presented in Figure 12.

- **ParameterDeletionResultResponse**

ParameterDeletionResultResponse is of complex type DeletionResultResponse, the structure of which is presented in Figure 15.

- **ParametersByToolIdResponse**

ParametersByToolIdResponse is of complex type ParametersResponse, the structure of which is presented in Figure 15.

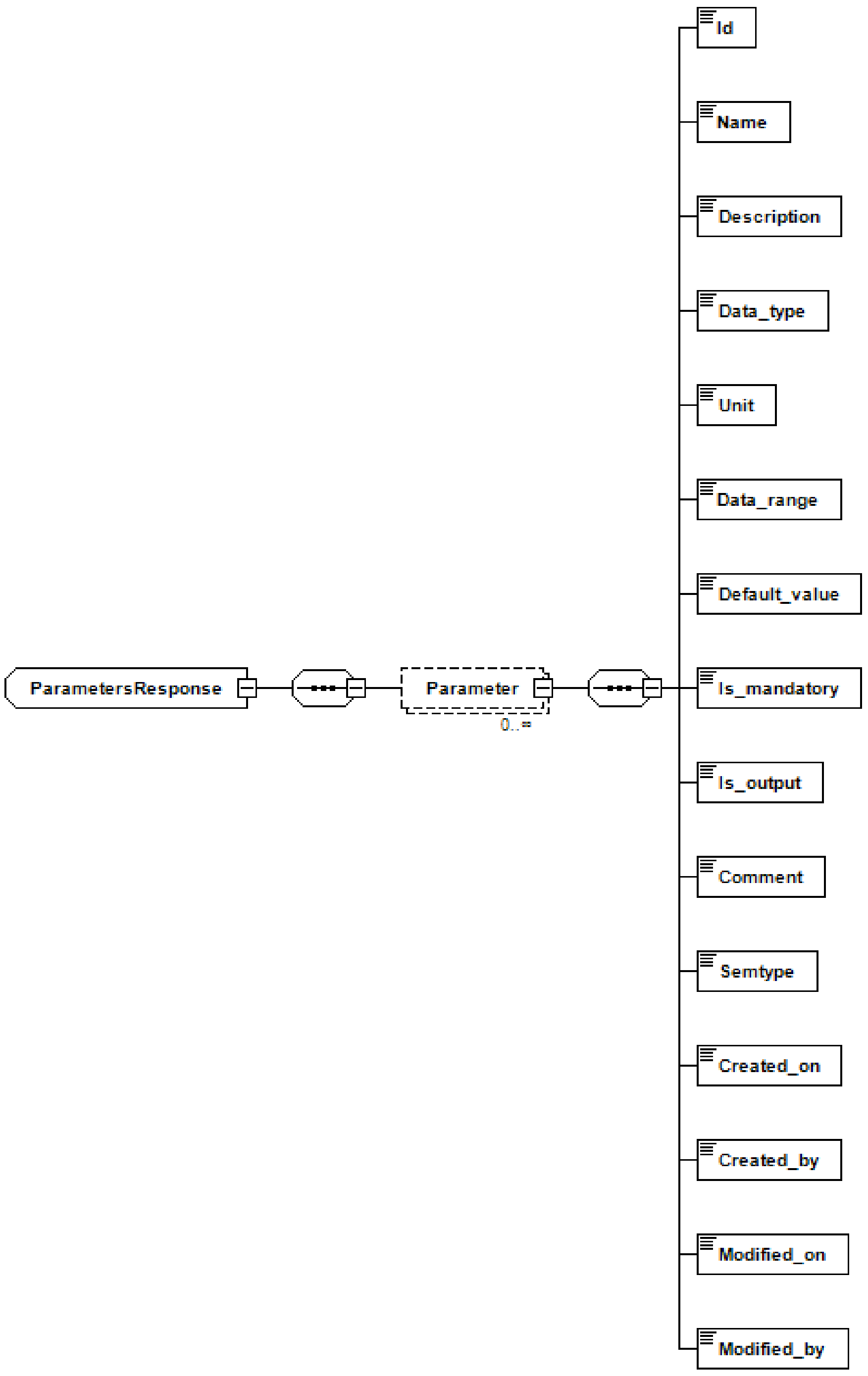


Figure 16 ParametersResponse Complex Type tree view

- **MandatoryParametersByToolIdResponse**

MandatoryParametersByToolIdResponse is of complex type ParametersResponse, the structure of which is presented in Figure 15.

- **InputParametersByToolIdResponse**

InputParametersByToolIdResponse is of complex type ParametersResponse, the structure of which is presented in Figure 14.

- **OutputParametersByToolIdResponse**

OutputParametersByToolIdResponse is of complex type ParametersResponse, the structure of which is presented in Figure 14.

Property

Method	Arguments	Returned object	Description
storeProperty	• String name • String description • String comment	PropertyIdResponse	This method stores the basic descriptive information of a property and returns the id. If the id already exists, updates the values
getAllProperties		AllPropertiesResponse	This method returns all the properties and the corresponded descriptive information stored (name, description, comment) and null when not existing
getPropertyById	• Int id	PropertyByIdResponse	This method returns the descriptive information stored under the property id (name, description, comment) and null when not existing
storePropertyValue	• Int tool_id • Int property_id • String value	PropertyValueIdResponse	This method stores the value of a property for a tool and returns the id. If the id already exists, updates the values
deletePropertyValue	• Int id	PropertyValueDeletionResultResponse	This method deletes the property value for a certain tool and returns the result Success or Fail
getPropertyValuesByToolId	• Int tool_id	PropertyValuesByToolIdResponse	This method retrieves all the property-value pairs for a given tool
deletePropertyById	• IntId	PropertyDeletionResultResponse	This method deletes the property of the given id and the corresponded values. Returns Success or Fail

- **PropertyIdResponse**

PropertyIdResponse is of complex type IdResponse, the structure of which is presented in Figure 12.

- **AllPropertiesResponse**

AllPropertiesResponse is of complex type PropertiesResponse, the structure of which is presented in Figure 17.

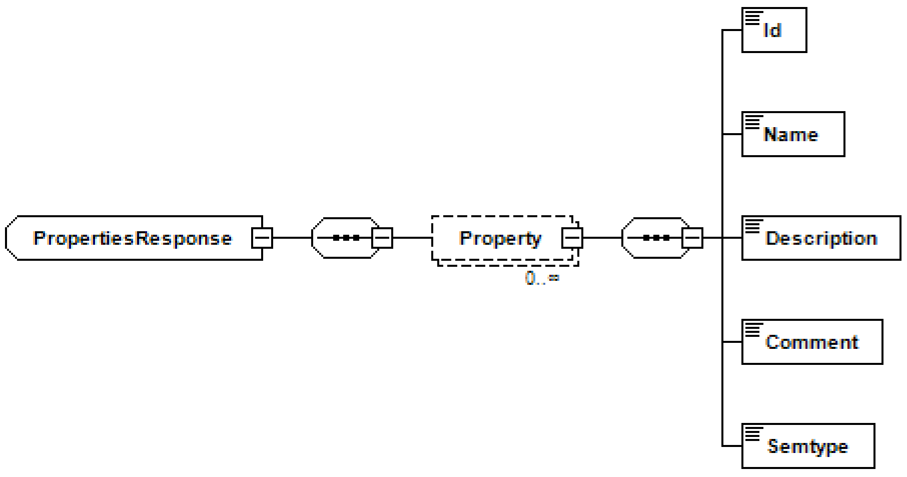


Figure 17 PropertiesResponse Complex Type tree view

- **PropertyByIdResponse**

PropertyByIdResponse is of complex type PropertyResponse, the structure of which is presented in Figure 17Figure 17.

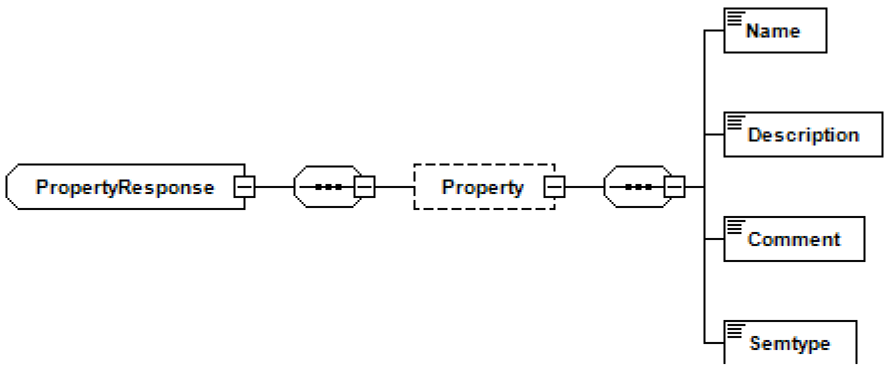


Figure 18 PropertyResponse Complex Type tree view

- **PropertyValueIdResponse**

PropertyValueIdResponse is of complex type IdResponse, the structure of which is presented in Figure 12.

- **PropertyValueDeletionResultResponse**

PropertyValueDeletionResultResponse is of complex type DeletionResultResponse, the structure of which is presented in Figure 15.

- **PropertyValuesByToolIdResponse**

PropertyValuesByToolIdResponse is of complex type PropertyValuesResponse, the structure of which is presented in Figure 19.

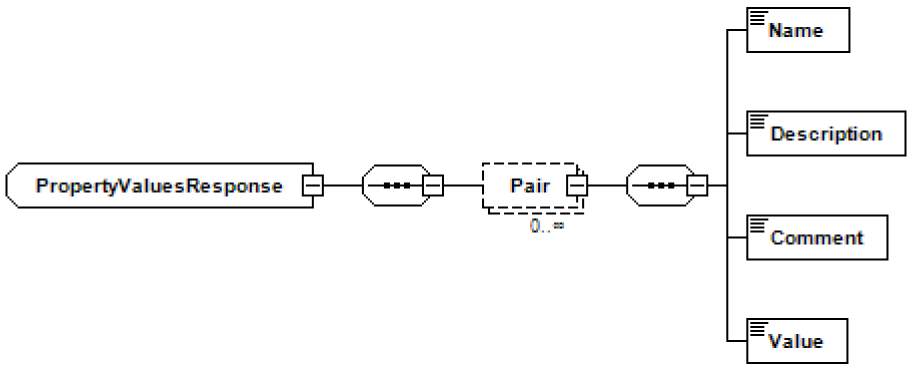


Figure 19 PropertyValuesResponse Complex Type tree view

- **PropertyDeletionResultResponse**

PropertyDeletionResultResponse is of complex type DeletionResultResponse, the structure of which is presented in Figure 15.

Reference

Method	Arguments	Returned object	Description
storeReference	• Int tool_id • String title • String type • String creator • String issued • String bibliographic_citation • String is_part_of • String source	ReferenceIdResponse	This method stores the reference information of the tool and returns the id. If the id already exists, updates the values
deleteReferenceById	• Int id	ReferenceDeletionResultResponse	This method deletes a specific reference and returns Success of Fail of the action
getReferencesByToolId	• Int tool_id	ReferencesByToolIdResponse	This method returns all the references of a given tool id and null when not existing
getAxes	• Int id • Options:	AxesResponse	This method returns all the references based on the

	String Ancestors/ String Descendants/ String Siblings • Optional for the Ancestors Descendant arguments: Int level		arguments. It makes use of the is_part_of attribute and given the option and the level (if the option is different than that of sibling) returns the desired references with all the reference information

- **ReferenceIdResponse**

ReferenceIdResponse is of complex type IdResponse, the structure of which is presented in Figure 12.

- **ReferenceDeletionResultResponse**

ReferenceDeletionResultResponse is of complex type DeletionResultResponse, the structure of which is presented in Figure 15.

- **ReferencesByToolIdResponse**

ReferencesByToolIdResponse is of complex type ReferencesByToolIdResponse, the structure of which is presented in Figure 20.

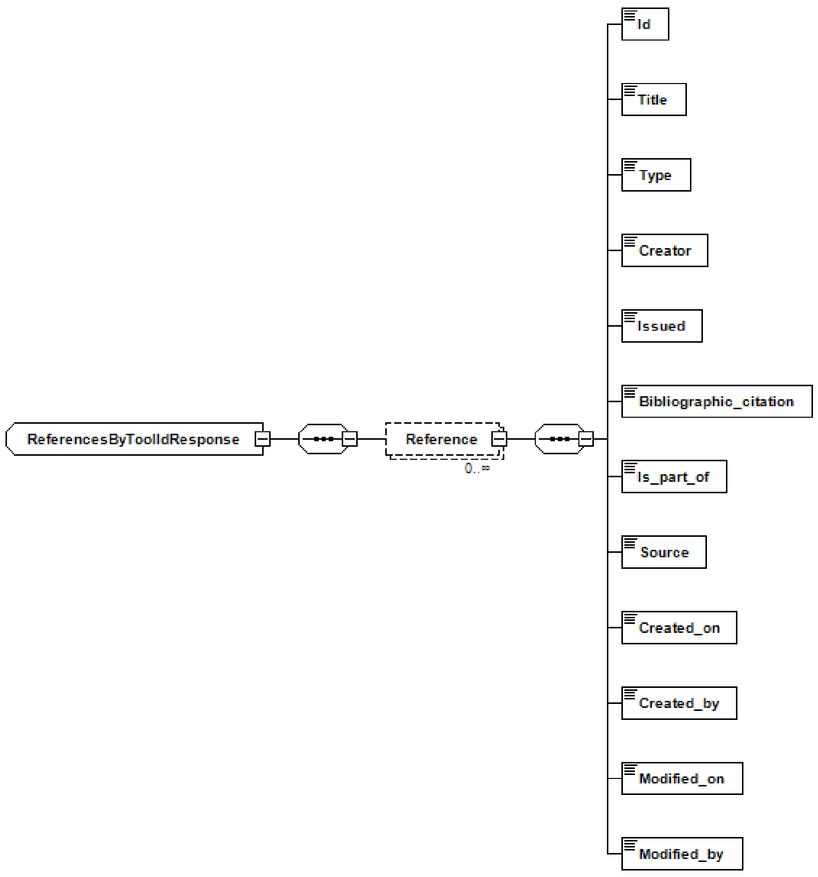


Figure 20 ReferencesByToolIdResponse Complex Type tree view

- **AxesResponse**

AxesResponse is of complex type ReferencesByToolIdResponse, the structure of which is presented in Figure 20.

File

Method	Arguments	Returned object	Description
storeFile	• Int id • String title • String description • String kind • String source • String version • String license • String sha1sum • String comment • String engine • Blob	FileIdResponse	This method stores the file information, persists the source to the source and returns the id. If the id already exists, updates the values
deleteFile	• Int id	FileDeletionResult Response	This method deletes a certain file and returns the result Success or Fail
getFileById	• Int id • Optional: String version	Blob	This method returns the file stored in the source and all the descriptive information
getLatestFilesByToolId	• Int tool_id	LatestFilesByToolIdResponse	This method returns the latest version and information of all the files of a given tool
getFilesOfKind	• Int tool_id • String kind	FilesOfKindResponse	This method returns the information of the latest version of the files of a specific kind
getPreviousVersions	• Int id	PreviousVersions Response	This method returns all the previous versions and information of a given file

- **FileIdResponse**

FileIdResponse is of complex type IdResponse, the structure of which is presented in Figure 12.

- **FileDeletionResultResponse**

FileDeletionResultResponse is of complex type DeletionResultResponse, the structure of which is presented in Figure 15.

- **LatestFilesByToolIdResponse**

LatestFilesByToolIdResponse is of complex type FilesResponse, the structure of which is presented in Figure 21.

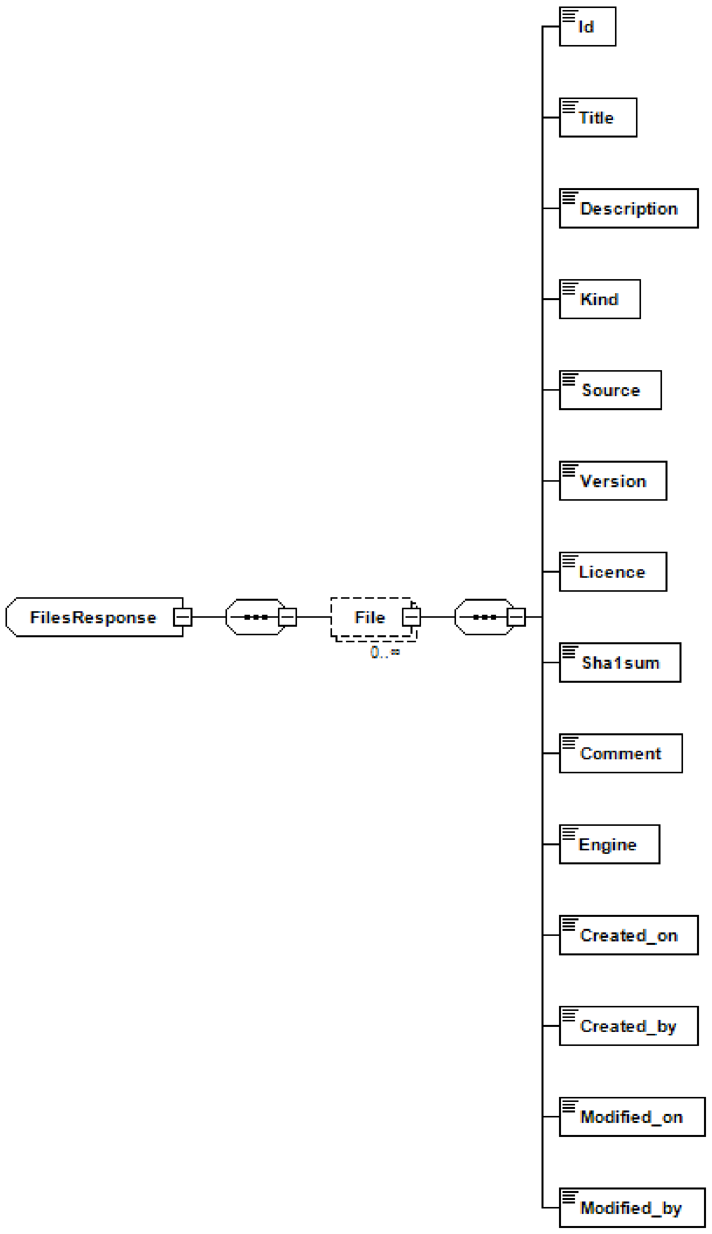


Figure 21 FilesResponse Complex Type tree view

- **FilesOfKindResponse**

FilesOfKindResponse is of complex type FilesResponse, the structure of which is presented in Figure 21.

- **PreviousVersionsResponse**

PreviousVersionsResponse is of complex type FilesResponse, the structure of which is presented in Figure 21.

Generic

Method	Arguments	Returned object	Description
searchByClassAttribute	• String class • String attribute • String query	SearchResultsResponse	This method creates a dynamic query from the arguments and returns the results found and null when not existing.

- **SearchResultsResponse**

SearchResultsResponse is of complex type SearchResultsResponse, the structure of which is presented in Figure 22.

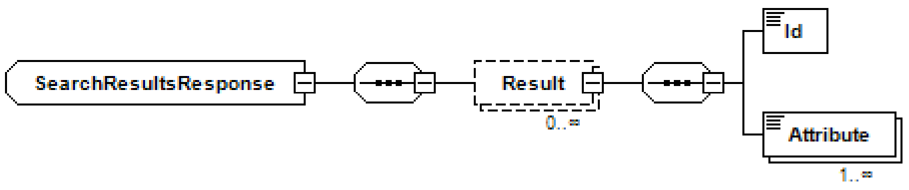


Figure 22 SearchResultsResponse Complex Type tree view

5.2 In Silico Trial Repository Service

This section includes the schema, an overview of the main entities and the design of the standardized interfaces of the In Silico Trial Repository.

5.2.1 In Silico Trial Repository Schema

The three main entities of the in Silico repository (Figure 23) are the trial, the experiment, and the subject. The trials are the digital equivalent of a clinical one, with models in place of the hypotheses and scientific methods, and placebo models in place of a placebo drug. For cancer cases a free growth model could be used as a placebo model.

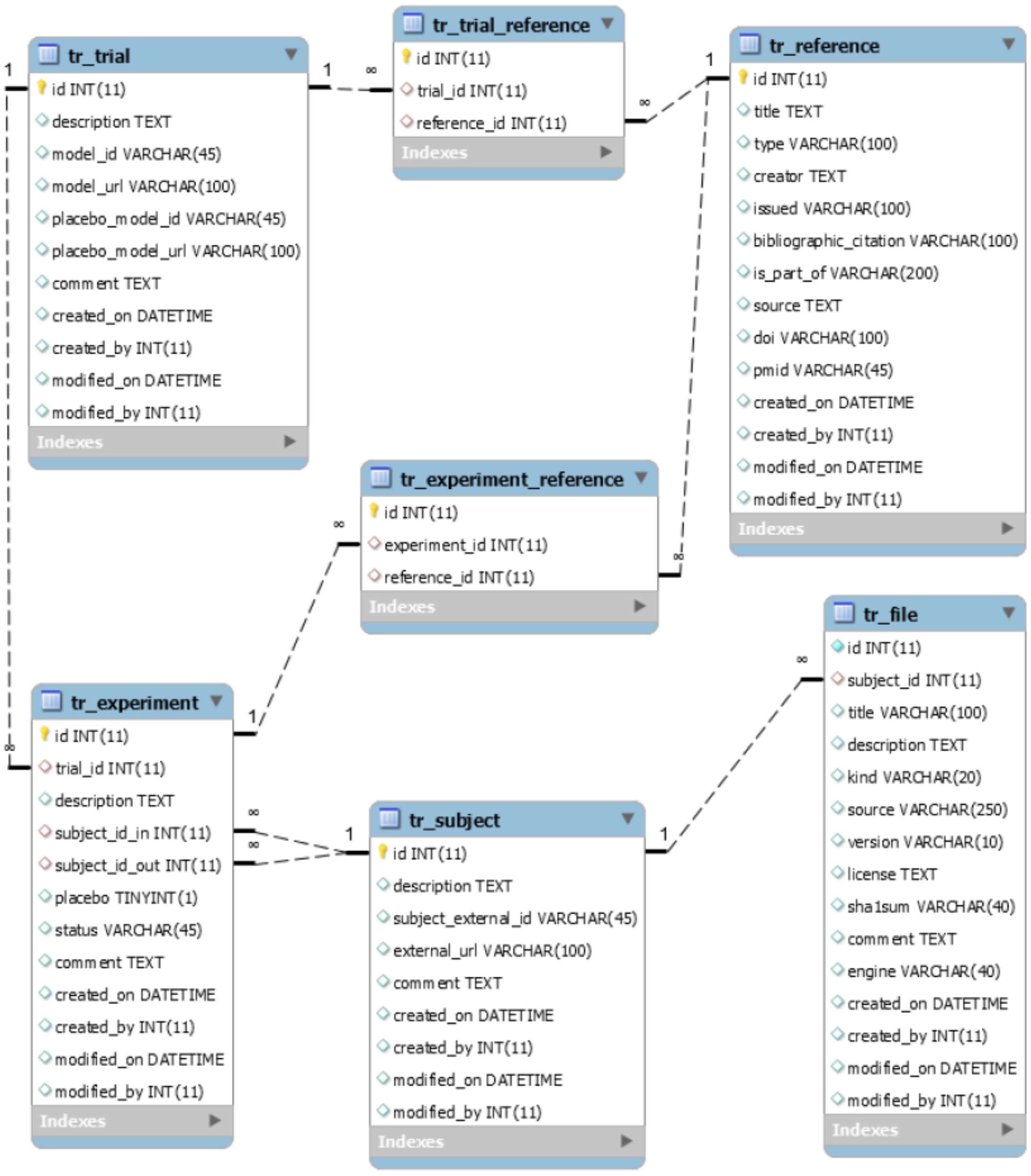


Figure 23 Entity-Relationship (ER) diagram of the in silico repository

A trial consists of multiple experiments. The experiments are performed with the same model each time which is defined in the referred trial. Each experiment scheduled or ran in the CHIC framework is stored in this repository in the form of triples. A triple consist of the status of the subject of the experiment before the experiment execution, the model that is stored in the trial table and the status of the subject after the experiment.

Each experiment is performed on a subject that is on a given state and creates a new subject with the new state. The external ids and URLs are referring to external repositories that could be used such as the CHIC clinical data system, a clinical trial management system (ObTiMA, OpenClinica), etc.

5.2.2 In Silico Trial Repository Interfaces

Trial

Method	Arguments	Returned object	Description
storeTrial	• String description • Int model_id • String model_url • Int placebo_model_id • String placebo_model_url • String comment	TrialIdResponse	This method stores the basic descriptive information of the trial, the model and placebo model. It returns the id of the trial. If the id already exists, updates the values
getAllTrials		AllTrialsResponse	This method returns all the trial ids, the corresponded descriptive information and models (id, descriptiption, model_id, model_url, placebo_model_id, placebo_model_url, comment). Returns null when not existing
getTrialById	• Int id	TrialByIdResponse	This method returns the descriptive information and the information of the models used in the trial, under the given id (descriptiption, model_id, model_url, placebo_model_id, placebo_model_url, comment). Returns null when not existing
getTrialsByModelId	• Int id	TrialsByModelIdResponse	This method returns the trials in which a model is used and the stored information under the id (id, descriptiption, model_id, model_url, placebo_model_id, placebo_model_url, comment).

			The argument is the id of the tool used in the model repository. It returns null when not existing
deleteTrialById	• Int id	TrialDeletionResultResponse	This method deletes the trial, the experiments included in the trial and the reference links. Returns Success or Fail

- **TrialIdResponse**

TrialIdResponse is of complex type IdResponse, the structure of which is presented in Figure 12.

- **AllTrialsResponse**

AllTrialsResponse is of complex type TrialsResponse, the structure of which is presented in Figure 24.

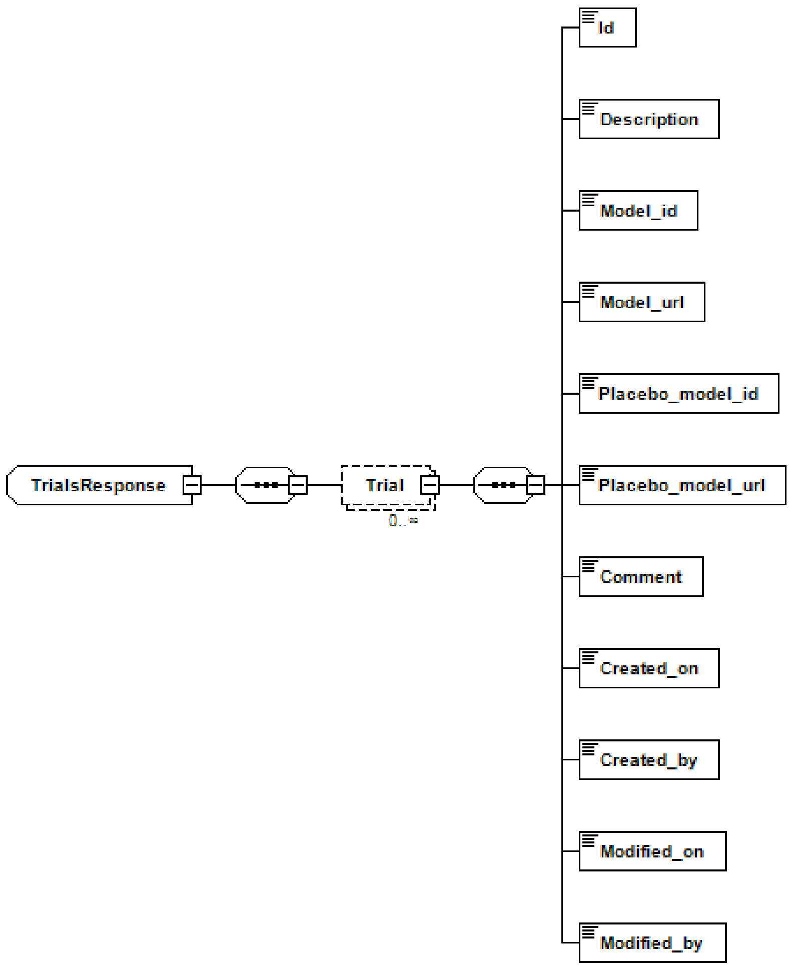


Figure 24 TrialsResponse Complex Type tree view

- **TrialByIdResponse**

TrialByIdResponse is of complex type TrialResponse, the structure of which is presented in Figure 25.

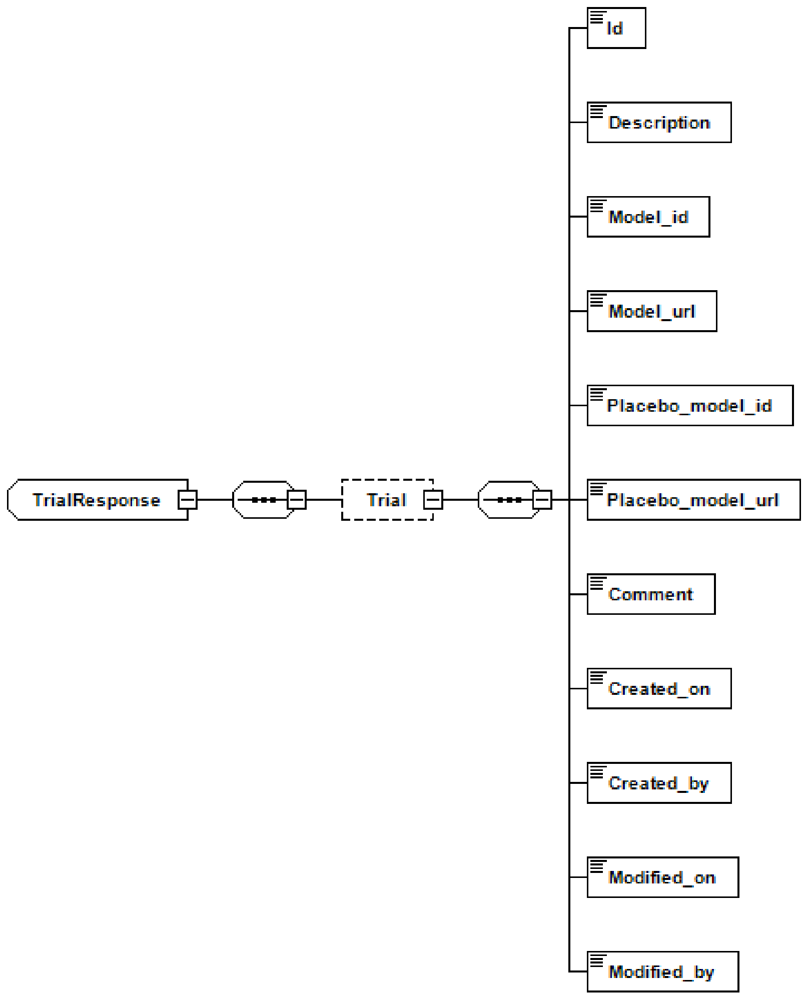


Figure 25 TrialResponse Comple Type tree view

- **TrialsByModelIdResponse**

TrialsByModelIdResponse is of complex type TrialsResponse, the structure of which is presented in Figure 24.

- **TrialDeletionResultResponse**

TrialDeletionResultResponse is of complex type DeletionResultResponse, the structure of which is presented in Figure 15.

Experiment

Method	Arguments	Returned object	Description
storeExperiment	• Int trial_id • String description • Int subject_id_in • Optional: Int subject_id_out • Bool placebo • String status • String comment	ExperimentIdResponse	This method stores the necessary and descriptive information of an experiment. It returns the id of the experiment. If the id already exists, updates the values
getAllExperimentsByTrialId	• Int trial_id	AllExperimentsByTrialIdResponse	This method returns all the experiments and the related information included in the trial (id, descriptiption, subject_id_in, subject_id_out, placebo, status, comment). Returns null when not existing
getExperimentById	• Int id	ExperimentByIdResponse	This method returns the experiment and the related information stored under the id (descriptiption, subject_id_in, subject_id_out, placebo, status, comment). Returns null when not existing
getExperimentStatusById	• Int id	ExperimentStatusByIdResponse	This method returns the status of an experiment
getExperimentsByStatus	• String status	ExperimentsByStatusResponse	This method returns all the experiments that are on a given status
updateExperimentStatus	• Int id • String status	ExperimentStatusResultResponse	This method updates the status of a given experiment. Returns Success or Fail
storeExperimentResult	• Int id • String status • Blob	ExperimentResultResponse	This method stores an intermediate or a final result of an experiment. It returns the id of the file. If the id already exists, updates the values
deleteExperimentById	• Int id	ExperimentDeletionResultResponse	This method deletes the experiment and the corresponded experiment references (links). Returns Success or Fail

- **ExperimentIdResponse**

ExperimentIdResponse is of complex type IdResponse, the structure of which is presented in Figure 12.

- **AllExperimentsByTrialIdResponse**

AllExperimentsByTrialIdResponse is of complex type ExperimentsResponse, the structure of which is presented in Figure 26.

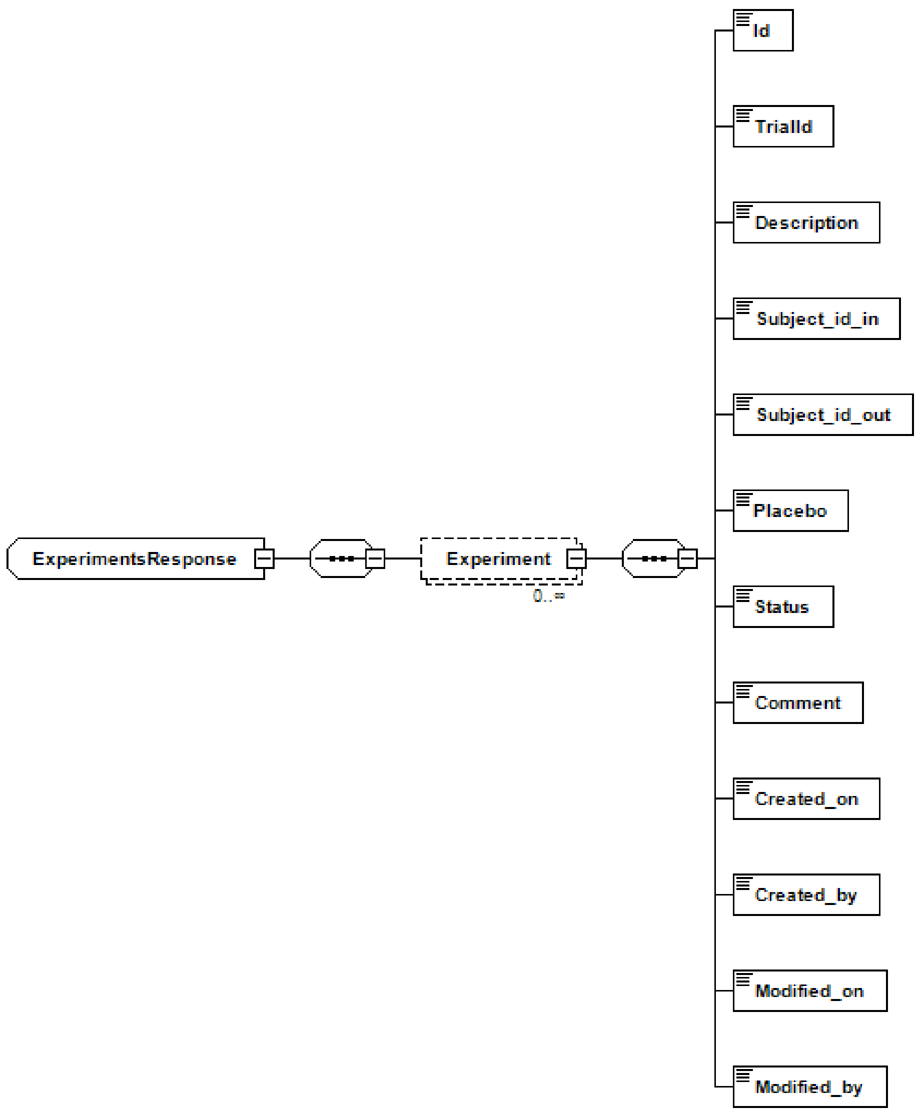


Figure 26 ExperimentsResponse Complex Type tree view

- **ExperimentByIdResponse**

ExperimentsByIdResponse is of complex type ExperimentResponse, the structure of which is presented in Figure 27.

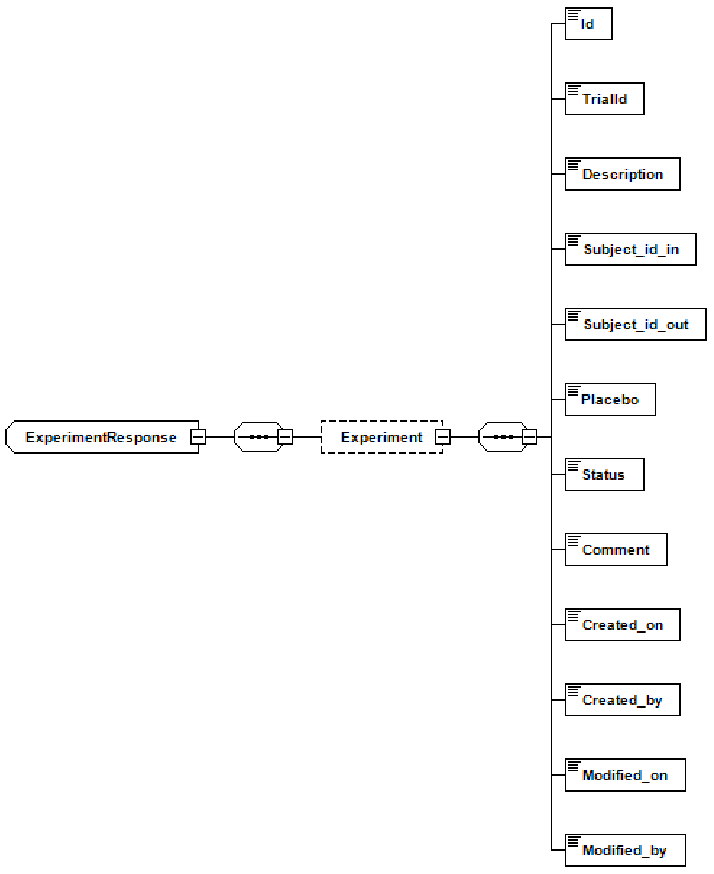


Figure 27 ExperimentResponse Complex Type tree view

- **ExperimentStatusByIdResponse**

ExperimentStatusByIdResponse is of complex type ExperimentStatusByIdResponse, the structure of which is presented in Figure 28Figure 26.

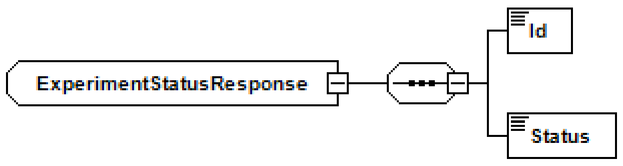


Figure 28 ExperimentStatusByIdResponse Complex Type tree view

- **ExperimentsByStatusResponse**

ExperimentsByStatusResponse is of complex type ExperimentsResponse, the structure of which is presented in Figure 26.

- **ExperimentStatusResultResponse**

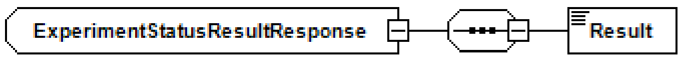


Figure 29 ExperimentStatusResultResponse Complex Type tree view

- **ExperimentResultResponse**

ExperimentResultResponse is of complex type IdResponse, the structure of which is presented in Figure 12.

- **ExperimentDeletionResultResponse**

ExperimentDeletionResultResponse is of complex type DeletionResultResponse, the structure of which is presented in Figure 15.

Subject

Method	Arguments	Returned object	Description
storeSubject	• String description • String subject_external_id • String external_url • String comment	SubjectIdResponse	This method stores information related to a subject. The method returns the id of the created subject. If the id already exists, updates the values
deleteSubjectById	• Int id	SubjectDeletionResultResponse	This method deletes a subject stored under the provided subject_id and the linked files. Returns Success or Fail
getAllSubjects		AllSubjectsResponse	This method returns all the subjects
getSubjectById	• Int id	SubjectByIdResponse	This method returns the subject and the related information stored under the id (description, subject_external_id, external_url, comment). Returns null when not existing

- **SubjectIdResponse**

SubjectIdResponse is of complex type IdResponse, the structure of which is presented in Figure 12.

- **SubjectDeletionResultResponse**

SubjectDeletionResultResponse is of complex type DeletionResultResponse, the structure of which is presented in Figure 15.

- **AllSubjectsResponse**

AllSubjectsResponse is of complex type SubjectsResponse, the structure of which is presented in Figure 30.

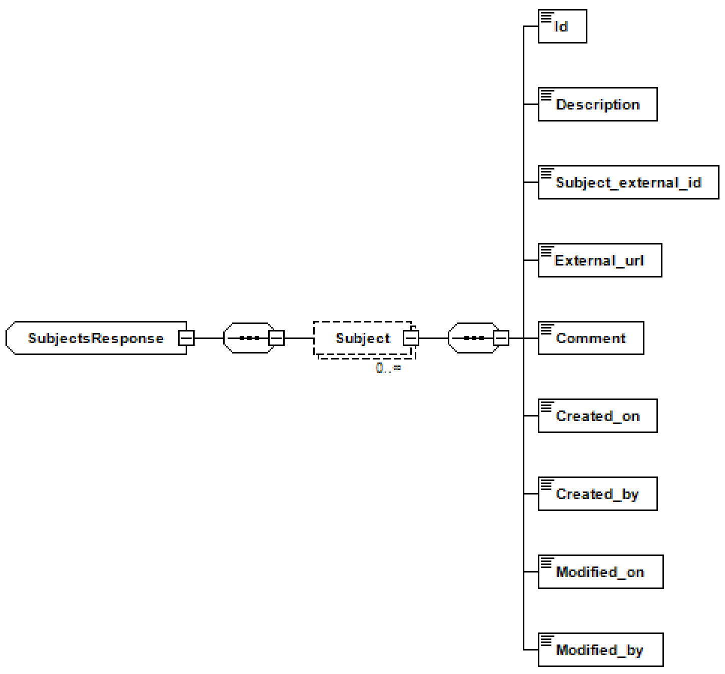


Figure 30 SubjectsResponse Complex Type tree view

- **SubjectByIdResponse**

SubjectByIdResponse is of complex type SubjectResponse, the structure of which is presented in Figure 31.

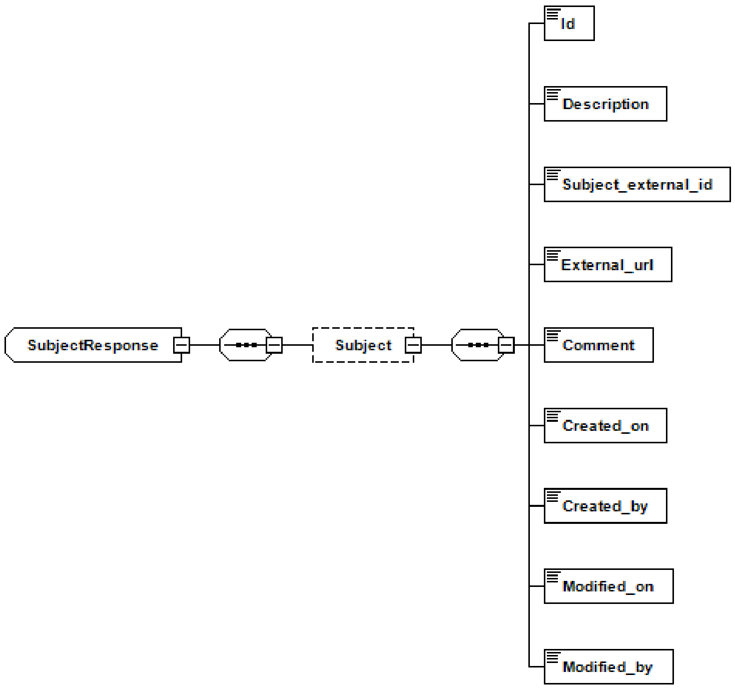


Figure 31 SubjectResponse Complex Type tree view

Reference

Method	Arguments	Returned object	Description
storeTrReference	• String title • String type • String creator • String issued • String bibliographic_citation • String is_part_of • String source • String doi • String pmid	TrReferenceIdResponse	This method stores the information of a reference and returns the id. If the id already exists, updates the values
getAllTrRefences		AllTrRefencesResponse	This method returns all the references and the related information and null when not existing
getTrReferencesByTrialId	• Int trial_id	TrReferencesByTrialIdResponse	This method returns all the references of the provided trial id and the related information.

			Returns null when not existing
getTrReferencesByExperimentId	• Int trial_id	TrReferencesByExperimentIdResponse	This method returns all the references of the provided experiment id and the related information. Returns null when not existing
deleteTrReferenceById	• Int id	TrReferenceDeletionResultResponse	This method deletes a reference and the corresponded links to trials or experiments and returns Success or Fail
storeLinkToReference	• Int reference_id • Options: experiment/trial • Int id	LinkToReferenceIdResponse	This method creates a link from a trial or an experiment to a reference. Returns the id of the link or null if the arguments do not correspond to a reference id and to experiment/trial id
deleteReferenceLinkById	• Int Id • Options: trial/ experiment	ReferenceDeletionResultResponse	This method deletes the reference of a trial or of an experiment depending on the provided argument
getTrAxes	• Int id • Options: String Ancestors/ String Descendants/ String Siblings • Optional for the Ancestors Descendant arguments: Int level	TrAxesResponse	This method returns all the references based on the arguments. It makes use of the is_part_of attribute and given the option and the level (if the option is different than that of sibling) returns the desired references with all the reference information

- **TrReferenceIdResponse**

TrReferenceIdResponse is of complex type IdResponse, the structure of which is presented in Figure 12.

- **AllTrRefencesResponse**

AllTrReferencesResponse is of complex type ReferencesResponse, the structure of which is presented in Figure 32.

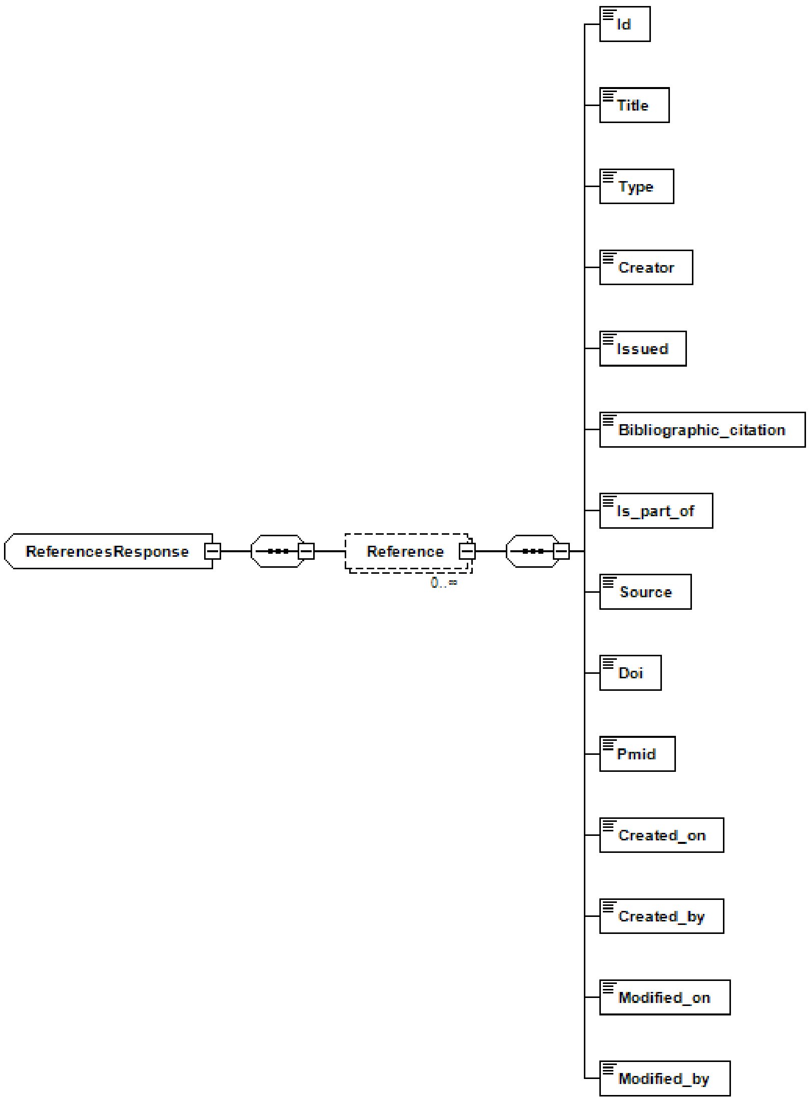


Figure 32 ReferencesResponse Complex Type tree view

- **TrReferencesByTrialIdResponse**

TrReferencesByTrialIdResponse is of complex type ReferencesResponse, the structure of which is presented in Figure 32.

- **TrReferencesByExperimentIdResponse**

TrReferencesByExperimentIdResponse is of complex type ReferencesResponse, the structure of which is presented in Figure 32.

- **TrReferenceDeletionResultResponse**

TrReferenceDeletionResultResponse is of complex type DeletionResultResponse, the structure of which is presented in Figure 15.

- **LinkToReferenceIdResponse**

LinkToReferenceIdResponse is of complex type IdResponse, the structure of which is presented in Figure 12.

- **ReferenceDeletionResultResponse**

ReferenceDeletionResultResponse is of complex type DeletionResultResponse, the structure of which is presented in Figure 15.

- **TrAxesResponse**

TrAxesResponse is of complex type ReferencesResponse, the structure of which is presented in Figure 32.

File

Method	Arguments	Returned object	Description
storeTrFile	• Int id • String title • String description • String kind • String source • String version • String sha1sum • String comment • Blob	TrFileIdResponse	This method stores the file information, persists the source to the source and returns the id. If the id already exists, updates the values
deleteTrFile	• Int id	TrFileDeletionResultResponse	This method deletes a certain file and returns the result Success or Fail
getTrFileById	• Int id • Optional: String version	Blob	This method returns the file stored in the source and all the descriptive information
getTrLatestFilesBySubjectId	• Int subject_id	TrLatestFilesBySubjectIdResponse	This method returns the latest version and information of all the files of a given tool
getTrFilesOfKind	• Int subject _id • String kind	TrFilesOfKindResponse	This method returns the information of the latest version of the files of a specific kind
getTrPreviousVersions	• Int id	TrPreviousVersionsResponse	This method returns all the previous versions and information of a given file

- **TrFileIdResponse**

TrFileIdResponse is of complex type IdResponse, the structure of which is presented in Figure 12.

- **TrFileDeletionResultResponse**

TrFileDeletionResultResponse is of complex type DeletionResultResponse, the structure of which is presented in Figure 15.

- **TrLatestFilesBySubjectIdResponse**

TrLatestFilesBySubjectIdResponse is of complex type FilesResponse, the structure of which is presented in Figure 33.

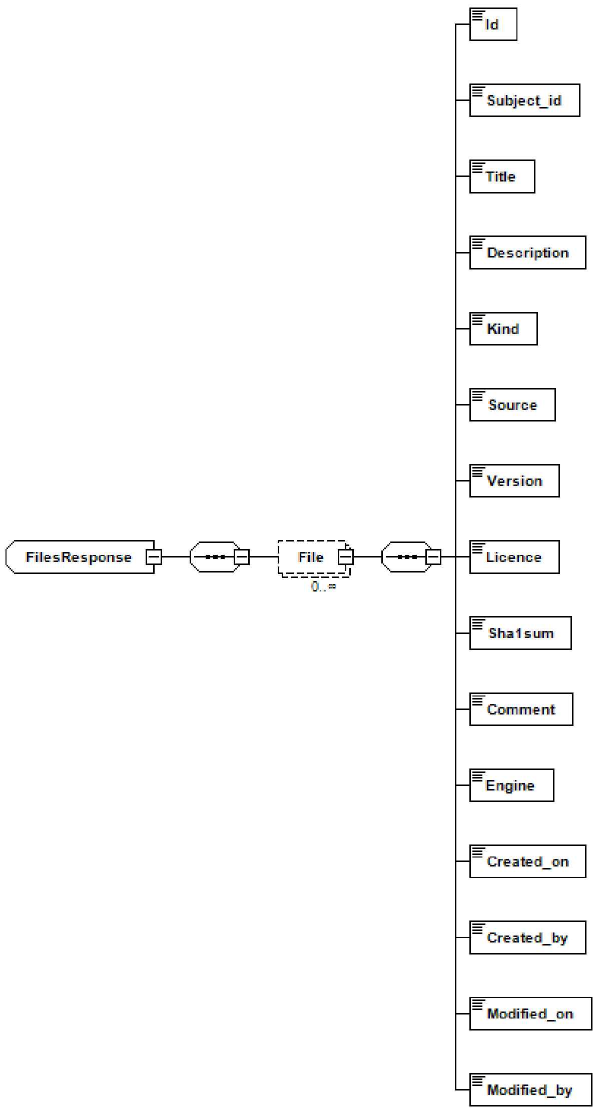


Figure 33 FilesResponse Complex Type tree view

- **TrFilesOfKindResponse**

TrFilesOfKindResponse is of complex type FilesResponse, the structure of which is presented in Figure 33.

TrPreviousVersionsResponse

TrPreviousVersionsResponse is of complex type FilesResponse, the structure of which is presented in Figure 33.

Generic

Method	Arguments	Returned object	Description
searchTrByClassAttribute	• String class • String attribute • String query	SearchTrResultsResponse	This method creates a dynamic query from the arguments and returns the results found and null when not existing.

- **SearchTrResultsResponse**

SearchTrResultsResponse is of complex type SearchResultsResponse, the structure of which is presented in Figure 22.

6 Use cases

In the following paragraphs we present two exemplary scenarios for the CHIC model orchestration platform. These use cases are in the core of the CHIC hypermodelling domain, namely the design of a new hypermodel and the execution of the hypermodels and retrieval of the results. For each scenario we present a sequence diagram, showing the components that interact and the messages that they exchange.

6.1 *Design of a new hypermodel*

Designing a new hypermodel starts with the user entering the Hypermodelling Editor, as shown in the sequence diagram below. The Editor needs to verify the authenticity of the user and therefore contacts the CHIC Authentication service, which is part of the security services. Of course in the case that the user has already logged into the CHIC platform (e.g. through the portal) then he/she does not need to provide his/her credentials (basically his/her user name and password) but there is still user validation going on behind the scenes to make user that this is a legitimate CHIC user.

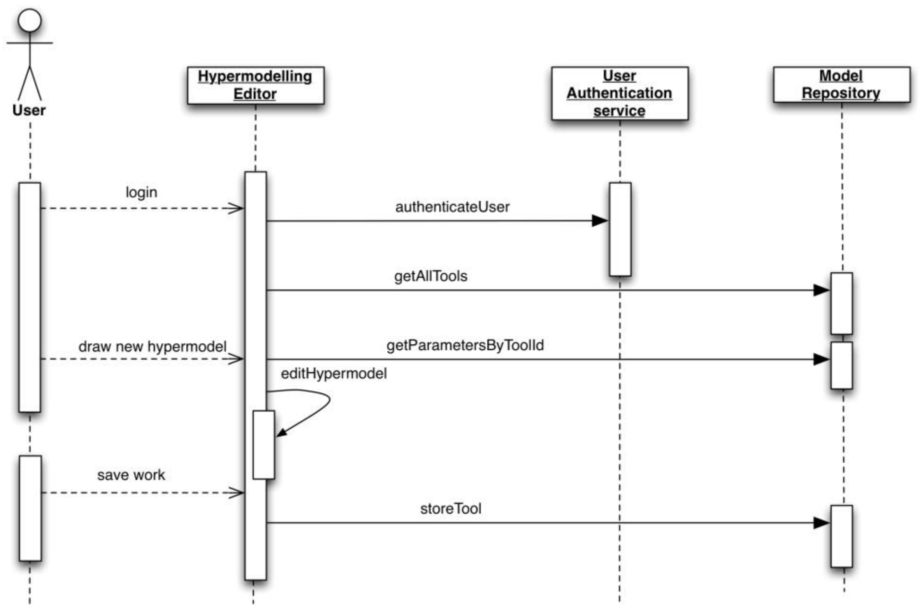


Figure 34 Design a new hypermodel sequence diagram

After the user's successful login, the Editor contacts the Model Repository and retrieves the available models using the "`getAllTools`" operation. For each of the selected hypomodels the user is then able to retrieve more details, for example information about their parameters (inputs and outputs) using the "`getParametersByToolId`" operation. Based on the retrieved hypomodels and their metadata, the user is then able to "draw" a new hypermodel by linking their inputs and outputs in a graphical way. Finally, the constructed hypermodel can be saved as a new "tool" in the Model Repository using the "`storeTool`" operation.

6.2 Hypermodel execution

The execution of a hypermodel is a more complex use case. The first complexity is the setup of a new *trial* and new *experiment* that is linked with the selected hypermodel. These concepts are described in Deliverable 8.1 but basically a trial binds together a specific hypermodel and a set of patient or other data ("subjects") that are either used as input or produced by the execution(s) of this hypermodel. Please refer to Figure 23 or to Figure 35 below for a view of this design. An experiment in this design corresponds to a single execution of the hypermodel using and producing some of the linked "subjects".

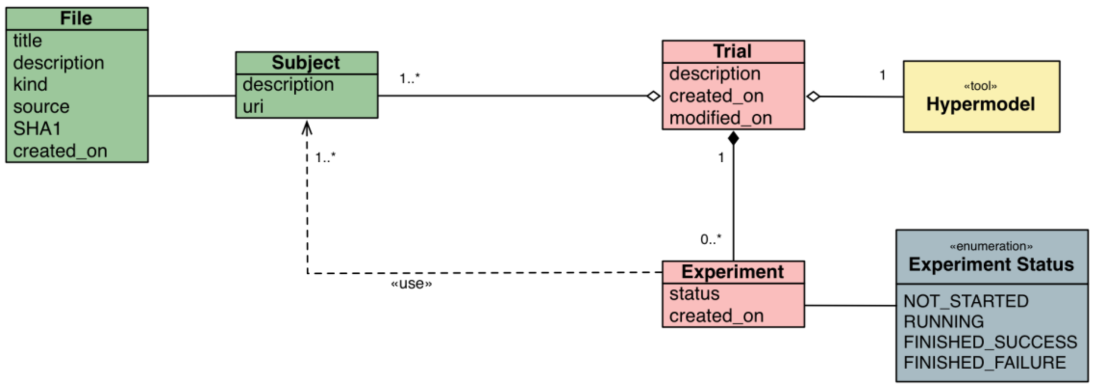


Figure 35 A simplified view of the trials-experiments design of the InSilico Trial repository

The second "complexity" (or interesting point) in this use case relates to the actual execution of the hypermodel. This includes the communication with the Hypermodelling Execution Framework, the retrieval of the model implementation details from the Model Repository, the retrieval and storage of "Experiment" related data in the InSilico Trial repository, the monitoring of the execution, and the presentation of the final results.

In more details, Figure 36 presents the components and their interactions during the submission of a hypermodel for execution. More specifically, this sequence diagram includes the following activities:

1. **Login**: The user logs into the Hypermodelling editor and the editor authenticates him/her by contacting the CHIC user authentication service. This is identical to the authentication process described in the previous paragraph.
2. **Hypermodel selection**: The Editor then retrieves the available models from the model repository and filters them so that it presents only the hypermodels[9] to the user. The user selects one of those hypermodels and the selected one is subsequently loaded.
3. **Trial and experiment creation**: The user is then guided to select one or more "subjects" to be used for the creation of new "trial" for the selected hypermodel. The subjects are retrieved using the "`getAllSubjects`" operation and the new trial is saved using the "`storeTrial`" method. After this, the user can create a new "experiment" for this trial, which means that it provides additional input parameters if needed, and the editor uses the "`storeExperiment`" operation to save it in the inSilico Trial repository.

[9] In the case where we have a large number of models the Editor can use the `searchByClassAttribute` advanced query interface so that only the models that are truly hypermodel are returned to the Editor.

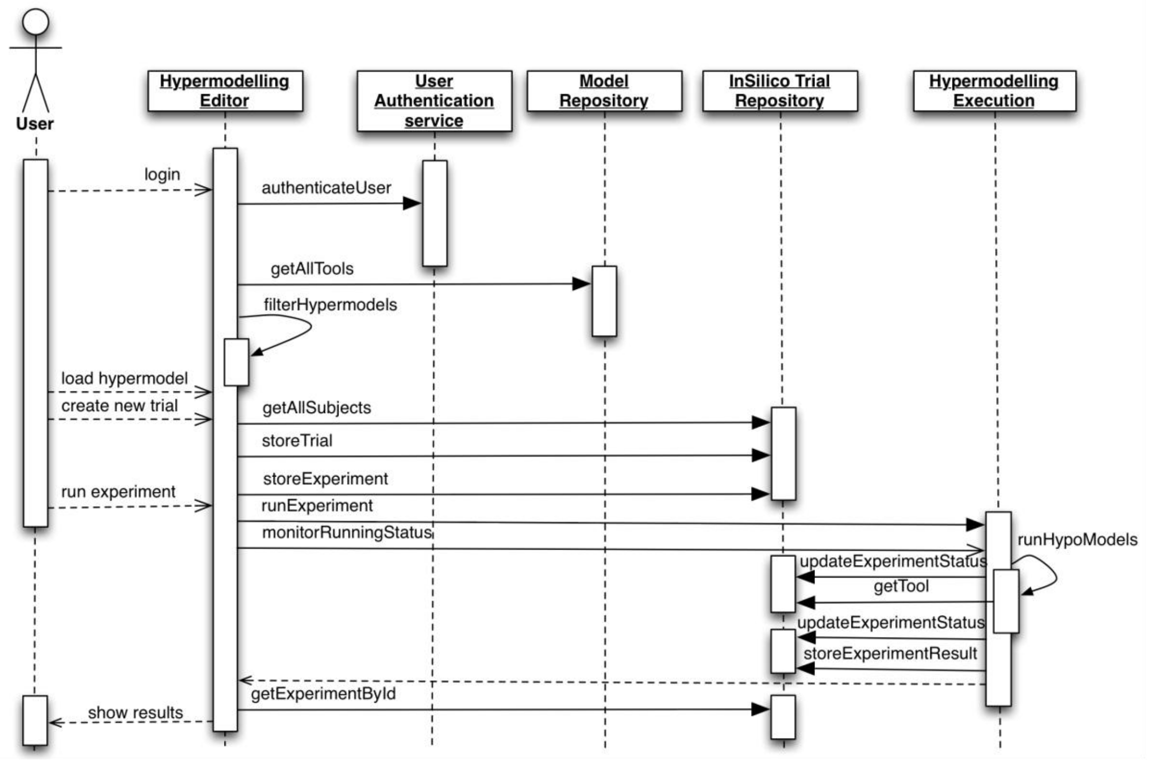


Figure 36 Hypermodel execution sequence diagram

4. **Hypermodel submission for execution**: On the press of a button, the user submits the hypermodel for execution in the Hypermodel Execution Framework of CHIC. The editor invokes a "`runExperiment`" operation and the Execution Framework starts gathering the needed information. The Editor also registers its "interest" in monitoring the status of the execution so that the user has an overview of the running progress.
5. **Hypermodel execution**: The execution framework retrieves the implementation details for each hypomodel included in the hypermodel by invoking the "`getTool`" and the related operations of the InSilico Trial repository. Additionally, it updates the status of the experiment (Figure 35) to "`RUNNING`" and finally to "`FINISHED_SUCCESS`" when all the hypomodels have completed their execution. The final results of the execution are then saved in the InSilico Trial repository using the "`storeExperimentResult`" operation.
6. **Presentation of results**: The Editor is notified about the successful termination of the hypermodel. It subsequently uses the "`getExperimentById`" to retrieve the complete information set for the specific experiment (i.e. hypermodel execution) which includes the produced files and results. Finally these results are shown to the user.

Appendix 1 – Abbreviations and acronyms

CHIC	*Computational Horizons in Cancer*
ER	*Entity Relationsship*
HF	*Hypermodel Framework*
HTML	*HyperText Markup Language*
RDF	*Resource Description Framework*
SPARQL	*Simple Protocol and RDF Query Language*
SSO	*Single Sign-On*
URL	*Uniform Resource Locator*
VPH	*Virtual Physiological Human*
WP	*Work Package*

Appendix 2 - XSD Schemas for the Model/Tool Repository Service

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
	<xs:element name="ToolIdResponse" type="IdResponse"/>
	<xs:element name="AllToolsResponse" type="AllToolsResponse"/>
	<xs:element name="ToolByIdResponse" type="ToolByIdResponse"/>
	<xs:element name="ToolDeletionResultResponse" type="DeletionResultResponse"/>
	<xs:element name="ParameterIdResponse" type="IdResponse"/>
	<xs:element name="ParameterDeletionResultResponse" type="DeletionResultResponse"/>
	<xs:element name="ParametersByToolIdResponse" type="ParametersResponse"/>
	<xs:element name="MandatoryParametersByToolIdResponse" type="ParametersResponse"/>
	<xs:element name="InputParametersByToolIdResponse" type="ParametersResponse"/>
	<xs:element name="OutputParametersByToolIdResponse" type="ParametersResponse"/>
	<xs:element name="PropertyIdResponse" type="IdResponse"/>
	<xs:element name="AllPropertiesResponse" type="PropertiesResponse"/>
	<xs:element name="PropertyByIdResponse" type="PropertyResponse"/>
	<xs:element name="PropertyValueIdResponse" type="IdResponse"/>
	<xs:element name="PropertyValueDeletionResultResponse" type="DeletionResultResponse"/>
	<xs:element name="PropertyValuesByToolIdResponse" type="PropertyValuesResponse"/>
	<xs:element name="PropertyDeletionResultResponse" type="DeletionResultResponse"/>
	<xs:element name="ReferenceIdResponse" type="IdResponse"/>
	<xs:element name="ReferenceDeletionResultResponse" type="DeletionResultResponse"/>
	<xs:element name="ReferencesByToolIdResponse" type="ReferencesByToolIdResponse"/>
	<xs:element name="AxesResponse" type="ReferencesByToolIdResponse"/>
	<xs:element name="FileIdResponse" type="IdResponse"/>
	<xs:element name="FileDeletionResultResponse" type="DeletionResultResponse"/>
	<xs:element name="LatestFilesByToolIdResponse" type="FilesResponse"/>
	<xs:element name="FilesOfKindResponse" type="FilesResponse"/>
	<xs:element name="PreviousVersionsResponse" type="FilesResponse"/>
	<xs:element name="SearchResultsResponse" type="SearchResultsResponse"/>
	<xs:element name="UpdateResultResponse" type="UpdateResultResponse"/>
	<xs:complexType name="IdResponse">
		<xs:sequence>
			<xs:element name="Id" type="xs:integer"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="DeletionResultResponse">
		<xs:sequence>
			<xs:element name="Result">
				<xs:simpleType>
					<xs:restriction base="xs:string">
						<xs:enumeration value="Success"/>
						<xs:enumeration value="Fail"/>
					</xs:restriction>
				</xs:simpleType>
			</xs:element>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="ToolByIdResponse">
		<xs:sequence minOccurs="0">
			<xs:element name="Title" type="xs:string"/>
			<xs:element name="Description" type="xs:long"/>
			<xs:element name="Comment" type="xs:long"/>
			<xs:element name="Created_on" type="xs:dateTime"/>
			<xs:element name="Created_by" type="xs:integer"/>
			<xs:element name="Modified_on" type="xs:dateTime"/>
			<xs:element name="Modified_by" type="xs:integer"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="AllToolsResponse">
		<xs:sequence>
			<xs:element name="Tool" minOccurs="0" maxOccurs="unbounded">
				<xs:complexType>
					<xs:sequence>
						<xs:element name="Title" type="xs:string"/>
						<xs:element name="Description" type="xs:long"/>
						<xs:element name="Comment" type="xs:long"/>
						<xs:element name="Created_on" type="xs:dateTime"/>
						<xs:element name="Created_by" type="xs:integer"/>
						<xs:element name="Modified_on" type="xs:dateTime"/>
						<xs:element name="Modified_by" type="xs:integer"/>
					</xs:sequence>
				</xs:complexType>
```

```
                </xs:element>
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="ParametersResponse">
            <xs:sequence>
                <xs:element name="Parameter" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="Id" type="xs:integer"/>
                            <xs:element name="Name" type="xs:string"/>
                            <xs:element name="Description" type="xs:long"/>
                            <xs:element name="Data_type" type="xs:string"/>
                            <xs:element name="Unit" type="xs:string"/>
                            <xs:element name="Data_range" type="xs:string"/>
                            <xs:element name="Default_value" type="xs:string"/>
                            <xs:element name="Is_mandatory" type="xs:integer"/>
                            <xs:element name="Is_output" type="xs:integer"/>
                            <xs:element name="Comment" type="xs:long"/>
                            <xs:element name="Semtype" type="xs:long"/>
                            <xs:element name="Created_on" type="xs:dateTime"/>
                            <xs:element name="Created_by" type="xs:integer"/>
                            <xs:element name="Modified_on" type="xs:dateTime"/>
                            <xs:element name="Modified_by" type="xs:integer"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="PropertiesResponse">
            <xs:sequence>
                <xs:element name="Property" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="Id" type="xs:integer"/>
                            <xs:element name="Name" type="xs:string"/>
                            <xs:element name="Description" type="xs:long"/>
                            <xs:element name="Comment" type="xs:long"/>
                            <xs:element name="Semtype" type="xs:long"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="PropertyResponse">
            <xs:sequence>
                <xs:element name="Property" minOccurs="0">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="Name" type="xs:string"/>
                            <xs:element name="Description" type="xs:long"/>
                            <xs:element name="Comment" type="xs:long"/>
                            <xs:element name="Semtype" type="xs:long"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="PropertyValuesResponse">
            <xs:sequence>
                <xs:element name="Pair" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="Name" type="xs:string"/>
                            <xs:element name="Description" type="xs:long"/>
                            <xs:element name="Comment" type="xs:long"/>
                            <xs:element name="Value" type="xs:long"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="ReferencesByToolIdResponse">
            <xs:sequence>
                <xs:element name="Reference" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="Id" type="xs:integer"/>
                            <xs:element name="Title" type="xs:long"/>
                            <xs:element name="Type" type="xs:string"/>
                            <xs:element name="Creator" type="xs:long"/>
                            <xs:element name="Issued" type="xs:string"/>
                            <xs:element name="Bibliographic_citation" type="xs:string"/>
                            <xs:element name="Is_part_of" type="xs:string"/>
                            <xs:element name="Source" type="xs:long"/>
                            <xs:element name="Created_on" type="xs:dateTime"/>
                            <xs:element name="Created_by" type="xs:integer"/>
                            <xs:element name="Modified_on" type="xs:dateTime"/>
                            <xs:element name="Modified_by" type="xs:integer"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="FilesResponse">
            <xs:sequence>
                <xs:element name="File" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="Id" type="xs:integer"/>
                            <xs:element name="Title" type="xs:string"/>
                            <xs:element name="Description" type="xs:long"/>
                            <xs:element name="Kind" type="xs:string"/>
                            <xs:element name="Source" type="xs:string"/>
                            <xs:element name="Version" type="xs:string"/>
                            <xs:element name="Licence" type="xs:long"/>
                            <xs:element name="Sha1sum" type="xs:string"/>
                            <xs:element name="Comment" type="xs:long"/>
                            <xs:element name="Engine" type="xs:string"/>
                            <xs:element name="Created_on" type="xs:dateTime"/>
                            <xs:element name="Created_by" type="xs:integer"/>
                            <xs:element name="Modified_on" type="xs:dateTime"/>
                            <xs:element name="Modified_by" type="xs:integer"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="SearchResultsResponse">
            <xs:sequence>
                <xs:element name="Result" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="Id" type="xs:integer"/>
                            <xs:element name="Attribute" type="xs:string"
    maxOccurs="unbounded"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="UpdateResultResponse">
            <xs:sequence>
                <xs:element name="Result">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="Success"/>
                            <xs:enumeration value="Fail"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
                <xs:element name="Id" type="xs:integer" minOccurs="0"/>
                <xs:element name="Attribute" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:schema>
```

Appendix 3 – XSD Schemas for the In Silico Trial Repository Service

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:element name="TrialIdResponse" type="IdResponse"/>
        <xs:element name="AllTrialsResponse" type="TrialsResponse"/>
        <xs:element name="TrialByIdResponse" type="TrialResponse"/>
        <xs:element name="TrialsByModelIdResponse" type="TrialsResponse"/>
        <xs:element name="TrialDeletionResultResponse" type="DeletionResultResponse"/>
        <xs:element name="ExperimentIdResponse" type="IdResponse"/>
        <xs:element name="AllExperimentsByTrialIdResponse" type="ExperimentsResponse"/>
        <xs:element name="ExperimentByIdResponse" type="ExperimentResponse"/>
        <xs:element name="ExperimentStatusByIdResponse" type="ExperimentStatusResponse"/>
        <xs:element name="ExperimentsByStatusResponse" type="ExperimentsResponse"/>
        <xs:element name="ExperimentStatusResultResponse" type="ExperimentStatusResultResponse"/>
        <xs:element name="ExperimentResultResponse" type="IdResponse"/>
        <xs:element name="ExperimentDeletionResultResponse" type="DeletionResultResponse"/>
        <xs:element name="SubjectIdResponse" type="IdResponse"/>
        <xs:element name="SubjectDeletionResultResponse" type="DeletionResultResponse"/>
        <xs:element name="AllSubjectsResponse" type="SubjectsResponse"/>
        <xs:element name="SubjectByIdResponse" type="SubjectResponse"/>
        <xs:element name="TrReferenceIdResponse" type="IdResponse"/>
        <xs:element name="AllTrRefencesResponse" type="ReferencesResponse"/>
        <xs:element name="TrReferencesByTrialIdResponse" type="ReferencesResponse"/>
        <xs:element name="TrReferencesByExperimentIdResponse" type="ReferencesResponse"/>
        <xs:element name="TrReferenceDeletionResultResponse" type="DeletionResultResponse"/>
        <xs:element name="LinkToReferenceIdResponse" type="IdResponse"/>
        <xs:element name="ReferenceDeletionResultResponse" type="DeletionResultResponse"/>
        <xs:element name="TrAxesResponse" type="ReferencesResponse"/>
        <xs:element name="TrFileIdResponse" type="IdResponse"/>
        <xs:element name="TrFileDeletionResultResponse" type="DeletionResultResponse"/>
        <xs:element name="TrLatestFilesBySubjectIdResponse" type="FilesResponse"/>
        <xs:element name="TrFilesOfKindResponse" type="FilesResponse"/>
        <xs:element name="TrPreviousVersionsResponse" type="FilesResponse"/>
        <xs:element name="SearchTrResultsResponse" type="SearchResultsResponse"/>
        <xs:element name="UpdateTrResultResponse" type="UpdateResultResponse"/>
        <xs:complexType name="IdResponse">
                <xs:sequence>
                        <xs:element name="Id" type="xs:integer"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="DeletionResultResponse">
                <xs:sequence>
                        <xs:element name="Result">
                                <xs:simpleType>
                                        <xs:restriction base="xs:string">
                                                <xs:enumeration value="Success"/>
                                                <xs:enumeration value="Fail"/>
                                        </xs:restriction>
                                </xs:simpleType>
                        </xs:element>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="TrialsResponse">
                <xs:sequence>
                        <xs:element name="Trial" minOccurs="0" maxOccurs="unbounded">
                                <xs:complexType>
                                        <xs:sequence>
                                                <xs:element name="Id" type="xs:integer"/>
                                                <xs:element name="Description" type="xs:long"/>
                                                <xs:element name="Model_id" type="xs:string"/>
                                                <xs:element name="Model_url" type="xs:string"/>
                                                <xs:element name="Placebo_model_id" type="xs:string"/>
                                                <xs:element name="Placebo_model_url"
type="xs:string"/>
                                                <xs:element name="Comment" type="xs:long"/>
                                                <xs:element name="Created_on" type="xs:dateTime"/>
                                                <xs:element name="Created_by" type="xs:integer"/>
                                                <xs:element name="Modified_on" type="xs:dateTime"/>
                                                <xs:element name="Modified_by" type="xs:integer"/>
                                        </xs:sequence>
                                </xs:complexType>
                        </xs:element>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="TrialResponse">
                <xs:sequence>
                        <xs:element name="Trial" minOccurs="0" maxOccurs="1">
                                <xs:complexType>
                                        <xs:sequence>
                                                <xs:element name="Id" type="xs:integer"/>
                                                <xs:element name="Description" type="xs:long"/>
                                                <xs:element name="Model_id" type="xs:string"/>
                                                <xs:element name="Model_url" type="xs:string"/>
                                                <xs:element name="Placebo_model_id" type="xs:string"/>
                                                <xs:element name="Placebo_model_url"
type="xs:string"/>

                                                <xs:element name="Comment" type="xs:long"/>
                                                <xs:element name="Created_on" type="xs:dateTime"/>
                                                <xs:element name="Created_by" type="xs:integer"/>
                                                <xs:element name="Modified_on" type="xs:dateTime"/>
                                                <xs:element name="Modified_by" type="xs:integer"/>
                                        </xs:sequence>
                                </xs:complexType>
                        </xs:element>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="ExperimentsResponse">
                <xs:sequence>
                        <xs:element name="Experiment" minOccurs="0" maxOccurs="unbounded">
                                <xs:complexType>
                                        <xs:sequence>
                                                <xs:element name="Id" type="xs:integer"/>
                                                <xs:element name="TrialId" type="xs:integer"/>
                                                <xs:element name="Description" type="xs:long"/>
                                                <xs:element name="Subject_id_in" type="xs:integer"/>
                                                <xs:element name="Subject_id_out" type="xs:integer"/>
                                                <xs:element name="Placebo" type="xs:integer"/>
                                                <xs:element name="Status" type="xs:string"/>
                                                <xs:element name="Comment" type="xs:long"/>
                                                <xs:element name="Created_on" type="xs:dateTime"/>
                                                <xs:element name="Created_by" type="xs:integer"/>
                                                <xs:element name="Modified_on" type="xs:dateTime"/>
                                                <xs:element name="Modified_by" type="xs:integer"/>
                                        </xs:sequence>
                                </xs:complexType>
                        </xs:element>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="ExperimentResponse">
                <xs:sequence>
                        <xs:element name="Experiment" minOccurs="0" maxOccurs="1">
                                <xs:complexType>
                                        <xs:sequence>
                                                <xs:element name="Id" type="xs:integer"/>
                                                <xs:element name="TrialId" type="xs:integer"/>
                                                <xs:element name="Description" type="xs:long"/>
                                                <xs:element name="Subject_id_in" type="xs:integer"/>
                                                <xs:element name="Subject_id_out" type="xs:integer"/>
                                                <xs:element name="Placebo" type="xs:integer"/>
                                                <xs:element name="Status" type="xs:string"/>
                                                <xs:element name="Comment" type="xs:long"/>
                                                <xs:element name="Created_on" type="xs:dateTime"/>
                                                <xs:element name="Created_by" type="xs:integer"/>
                                                <xs:element name="Modified_on" type="xs:dateTime"/>
                                                <xs:element name="Modified_by" type="xs:integer"/>
                                        </xs:sequence>
                                </xs:complexType>
                        </xs:element>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="SubjectResponse">
                <xs:sequence>
                        <xs:element name="Subject" minOccurs="0" maxOccurs="1">
                                <xs:complexType>
                                        <xs:sequence>
                                                <xs:element name="Id" type="xs:integer"/>
                                                <xs:element name="Description" type="xs:long"/>
                                                <xs:element name="Subject_external_id"
type="xs:string"/>
                                                <xs:element name="External_url" type="xs:string"/>
                                                <xs:element name="Comment" type="xs:long"/>
                                                <xs:element name="Created_on" type="xs:dateTime"/>
                                                <xs:element name="Created_by" type="xs:integer"/>
                                                <xs:element name="Modified_on" type="xs:dateTime"/>
                                                <xs:element name="Modified_by" type="xs:integer"/>
                                        </xs:sequence>
                                </xs:complexType>
                        </xs:element>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="SubjectsResponse">
                <xs:sequence>
                        <xs:element name="Subject" minOccurs="0" maxOccurs="unbounded">
                                <xs:complexType>
                                        <xs:sequence>
                                                <xs:element name="Id" type="xs:integer"/>
                                                <xs:element name="Description" type="xs:long"/>
                                                <xs:element name="Subject_external_id"
type="xs:string"/>
                                                <xs:element name="External_url" type="xs:string"/>
                                                <xs:element name="Comment" type="xs:long"/>
                                                <xs:element name="Created_on" type="xs:dateTime"/>
                                                <xs:element name="Created_by" type="xs:integer"/>
                                                <xs:element name="Modified_on" type="xs:dateTime"/>
                                                <xs:element name="Modified_by" type="xs:integer"/>
                                        </xs:sequence>
                                </xs:complexType>
                        </xs:element>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="ReferencesResponse">
                <xs:sequence>
                        <xs:element name="Reference" minOccurs="0" maxOccurs="unbounded">
                                <xs:complexType>
                                        <xs:sequence>
                                                <xs:element name="Id" type="xs:integer"/>
                                                <xs:element name="Title" type="xs:long"/>
                                                <xs:element name="Type" type="xs:string"/>
                                                <xs:element name="Creator" type="xs:long"/>
                                                <xs:element name="Issued" type="xs:string"/>
                                                <xs:element name="Bibliographic_citation"
type="xs:string"/>
                                                <xs:element name="Is_part_of" type="xs:string"/>
                                                <xs:element name="Source" type="xs:long"/>
                                                <xs:element name="Doi" type="xs:string"/>
                                                <xs:element name="Pmid" type="xs:string"/>
                                                <xs:element name="Created_on" type="xs:dateTime"/>
                                                <xs:element name="Created_by" type="xs:integer"/>
                                                <xs:element name="Modified_on" type="xs:dateTime"/>
                                                <xs:element name="Modified_by" type="xs:integer"/>
                                        </xs:sequence>
                                </xs:complexType>
                        </xs:element>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="FilesResponse">
                <xs:sequence>
                        <xs:element name="File" minOccurs="0" maxOccurs="unbounded">
                                <xs:complexType>
                                        <xs:sequence>
                                                <xs:element name="Id" type="xs:integer"/>
                                                <xs:element name="Subject_id" type="xs:integer"/>
                                                <xs:element name="Title" type="xs:string"/>
                                                <xs:element name="Description" type="xs:long"/>
                                                <xs:element name="Kind" type="xs:string"/>
                                                <xs:element name="Source" type="xs:string"/>
                                                <xs:element name="Version" type="xs:string"/>
                                                <xs:element name="Licence" type="xs:long"/>
                                                <xs:element name="Sha1sum" type="xs:string"/>
                                                <xs:element name="Comment" type="xs:long"/>
                                                <xs:element name="Engine" type="xs:string"/>
                                                <xs:element name="Created_on" type="xs:dateTime"/>
                                                <xs:element name="Created_by" type="xs:integer"/>
                                                <xs:element name="Modified_on" type="xs:dateTime"/>
                                                <xs:element name="Modified_by" type="xs:integer"/>
                                        </xs:sequence>
                                </xs:complexType>
                        </xs:element>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="SearchResultsResponse">
                <xs:sequence>
                        <xs:element name="Result" minOccurs="0" maxOccurs="unbounded">
                                <xs:complexType>
                                        <xs:sequence>
                                                <xs:element name="Id" type="xs:integer"/>
                                                <xs:element name="Attribute" type="xs:string"
maxOccurs="unbounded"/>
                                        </xs:sequence>
                                </xs:complexType>
                        </xs:element>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="UpdateResultResponse">
                <xs:sequence>
                        <xs:element name="Result">
                                <xs:simpleType>
                                        <xs:restriction base="xs:string">
                                                <xs:enumeration value="Success"/>
                                                <xs:enumeration value="Fail"/>
                                        </xs:restriction>
                                </xs:simpleType>
                        </xs:element>
                        <xs:element name="Id" type="xs:integer" minOccurs="0"/>
                        <xs:element name="Attribute" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="ExperimentStatusResponse">
                <xs:sequence>
                        <xs:element name="Id" type="xs:integer"/>
                        <xs:element name="Status" type="xs:string"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="ExperimentStatusResultResponse">
                <xs:sequence>
                        <xs:element name="Result">
                                <xs:simpleType>
                                        <xs:restriction base="xs:string">
                                                <xs:enumeration value="Success"/>
                                                <xs:enumeration value="Fail"/>
                                        </xs:restriction>
                                </xs:simpleType>
                        </xs:element>
                </xs:sequence>
        </xs:complexType>
</xs:schema>
```