A Modular Repository-based Infrastructure for Simulation Model Storage and Execution Support in the Context of *In Silico* Oncology and *In Silico* Medicine

Nikolaos A. Christodoulou, Nikolaos E. Tousert, Eleni Ch. Georgiadi, Katerina D. Argyri, Fay D. Misichroni and Georgios S. Stamatakos

In Silico Oncology and *In Silico* Medicine Group, Institute of Communication and Computer Systems, National Technical University of Athens, Zografos, Greece.

ABSTRACT: The plethora of available disease prediction models and the ongoing process of their application into clinical practice – following their clinical validation – have created new needs regarding their efficient handling and exploitation. Consolidation of software implementations, descriptive information, and supportive tools in a single place, offering persistent storage as well as proper management of execution results, is a priority, especially with respect to the needs of large healthcare providers. At the same time, modelers should be able to access these storage facilities under special rights, in order to upgrade and maintain their work. In addition, the end users should be provided with all the necessary interfaces for model execution and effortless result retrieval. We therefore propose a software infrastructure, based on a tool, model and data repository that handles the storage of models and pertinent execution-related data, along with functionalities for execution management, communication with third-party applications, user-friendly interfaces to access and use the infrastructure with minimal effort and basic security features.

KEYWORDS: *in silico* oncology, repository, database, model execution platform, Oncosimulator, multiscale cancer modeling

CITATION: Christodoulou et al. A Modular Repository-based Infrastructure for Simulation Model Storage and Execution Support in the Context of *In Silico* Oncology and *In Silico* Medicine. *Cancer Informatics* 2016:15 219–235 doi: 10.4137/CIN.S40189.

TYPE: Original Research

RECEIVED: May 24, 2016. **RESUBMITTED:** September 04, 2016. **ACCEPTED FOR PUBLICATION:** September 09, 2016.

ACADEMIC EDITOR: J. T. Efird, Editor in Chief

PEER REVIEW: Four peer reviewers contributed to the peer review report. Reviewers' reports totaled 724 words, excluding any confidential comments to the academic editor.

FUNDING: This work has been supported in part by the European Commission under the project MyHealthAvatar: a Demonstration of 4D Digital avatar Infrastructure for Access of Complete Patient Information (fP7-ICt-2011-9-600929), and Computational Horizons in Cancer (CHIC): Developing meta- and Hyper-multiscale models and repositories for *In Silico* oncology (fP7-ICt-2011-9, Grant agreement no: 600841). The authors confirm that the funder had no influence over the study design, content of the article, or selection of this journal.

COMPETING INTERESTS: Authors disclose no potential conflicts of interest.

CORRESPONDENCE: gestam@mail.ntua.gr

Introduction

As the concept of personalized medicine[1,2] is being increasingly prioritized by the clinical community, the need for software tools that could tailor treatment according to the patient's individual data is becoming more and more demanding. This reality has stimulated the development of mathematical and computational simulation models of disease development and response to treatment through various frameworks such as the virtual physiological human (VPH) initiative.[3] In the case of cancer, numerous mechanisms are involved at different biocomplexity scales. This dictates the adoption of multiscale approaches for the simulation of related phenomena.[4] Appropriate reuse of already existing elementary biomodels has led to the creation of complex and highly refined models known as hypermodels.[5]

A long-term goal of *in silico* or computational medicine is to provide clinicians with as many validated and clinically meaningful models as possible so that they can be efficiently supported in their decision-making procedure. Achieving such a goal involves a large number of stakeholders at different time points of a model life cycle. Modelers need to store their work in a facility that is independent of the development kit that has been used to implement the source code. The facility should also allow effortless testing, validation and subsequent update, and maintenance of the stored models as needed. Clinicians should have access to the functionalities provided. This includes methods to introduce the necessary input data to the system, either directly by completing forms with certain required values or by retrieving the data from third-party sources. Furthermore, large health and research institutions usually require all the aforementioned services to be consolidated into a single integrated solution,[6] thereby facilitating their use as well as the dissemination and the exploitation of the generated results.

In this paper, we propose a software infrastructure based on a repository that allows the involved stakeholders to perform the required tasks through a number of independent and interconnected modules. The paper starts with a conceptual high-level description of the system developed. Subsequently, it outlines each system component by also providing the component interrelations. The two implementation paradigms concerning nephroblastoma (Wilms tumor) and breast cancer are

discussed in the next section. The "Results" section provides exemplary outputs of the infrastructure for two different user cases considered, namely, for a clinical user's workflow and for a modeler's workflow. Both nephroblastoma and breast cancer have been addressed by both workflows. Indicative performance data are also included. The paper concludes with a discussion on the functionalities, the predictions, and the performance characteristics of the infrastructure including future extensions.

Conceptual System Overview

The primary idea, on which the present infrastructure has been built, is based on a set of relational databases. Aiming to ensure the enforcement of the Atomicity, Consistency, Isolation, Durability (ACID) principles and the handling of properly formed data, the database set is *wrapped* by a *shell*, which includes all communication methods and procedures between the infrastructure and the stakeholder groups, as well as all input data validation controls. Around this shell lays a second one, related to the security of the entire system.

To achieve a level of standardization of the stored data descriptive information, a suitable schema had to be selected for the relational databases. According to this schema, for each stored object, its individual characteristics and structural elements are placed in different tables. One table handles one specific feature for all stored objects. To represent a stored object in full, these tables are linked together with one-to-many or many-to-many foreign key-based relations.

Figure 1 shows the high-level conceptual diagram of the envisaged system. This initial blueprint has been appropriately modified and extended, taking into account the types of the prospective users and their particular needs for communicating with the infrastructure. These modifications have been developed to satisfy a set of user requirements, which was introduced during the early stages of the infrastructure implementation.

Components and Interrelations

Infrastructure "Core". The infrastructure backbone is based on the model-view-controller (MVC)[7] architectural pattern. Each data storage unit – in this case, a database table – is assigned to a model. All available functions for model manipulation are accessed by the users through the controller methods. These include modifications of the database contents, tool executions, application programming interface (API) calls, return of corresponding results, and communication with the external engines (metadata and execution). The controller methods are accessible via a URL system. This dictates the operation of the views that correspond to the graphical user interfaces (GUIs). The views are provided to the users for the subset of the model-manipulating functions, which require manual input, such as uploading a tool. The application of the MVC pattern to the conceptual system diagram of Figure 1 results in the more specific infrastructure architecture depicted in Figure 2.

Repository databases. This is the part where the information that is managed by the infrastructure resides. It consists of at least two relational databases. One of them includes the information that pertains to the simulation models and additional software tools that may assist in the model execution (both item types herein after referred to as *tools* for the sake of generality). The entity–relationship schema, which is adopted, groups and separates the various characteristics and structural components of a tool, in a manner that is unaffected by the tool format, functionality, or simulated condition. Any data used in tool executions are placed in the second database, which may also be used as storage for execution results. Depending on the desired implementation scale, additional tables, associated with the operation of other infrastructure parts, can be grouped on a separate database or be included into the data repository on an ad hoc basis. A typical example is the use of a table as a log for tool executions, which can be fed by the execution engine.

To create the schema for the repository databases, a number of assumptions are made in order to achieve the conceptual separation of a tool's characteristics. Each tool has basic descriptive information, which is stored separately from its implementations and other basic features. One or more properties further describe and/or classify a tool. These properties provide additional information about the tool and may relate to its operating principle. The descriptive information or a property and its value per tool are stored in different tables. A property value is connected to the tool it characterizes using a many-to-many relationship with a dedicated table, thereby facilitating the reuse of the property in several tools. Furthermore, each tool has a number of parameters, which are divided into input parameters and output parameters. It is also accompanied by a set of files that include the tool implementations, documentation and instructions, supplementary scripts, etc. The database holds all the descriptive information, whereas

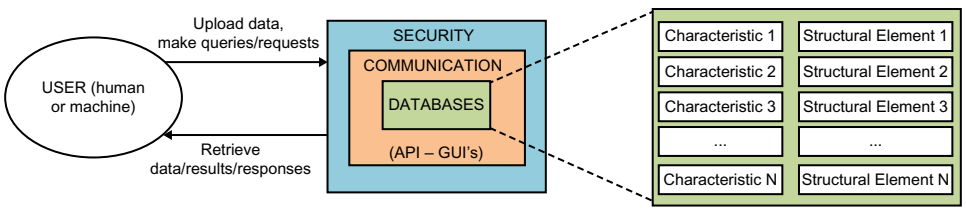


Figure 1. Conceptual diagram of the proposed architecture and database schema.

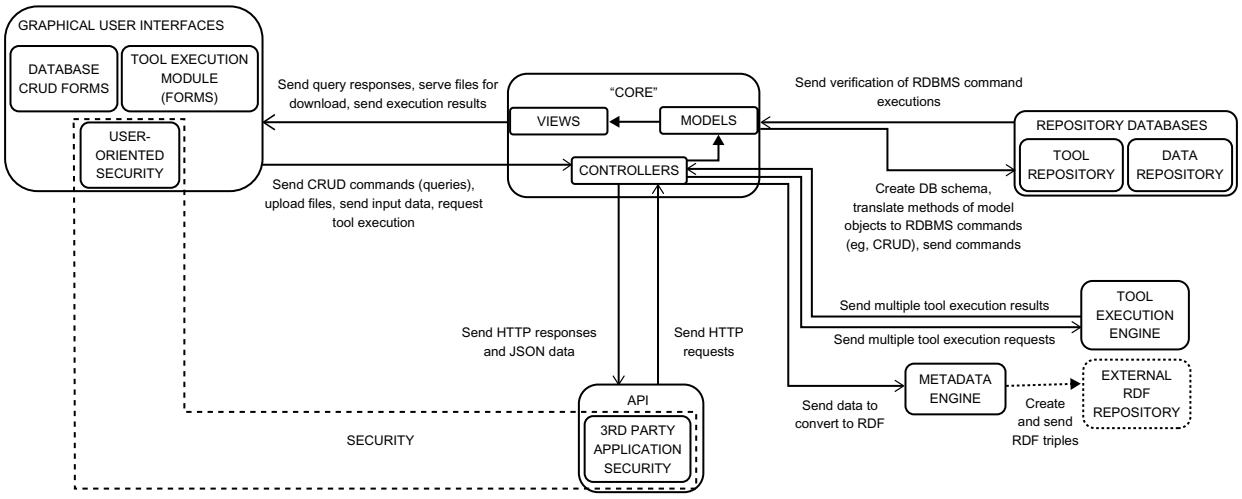


Figure 2. Architecture infrastructure.

the actual files (data) are stored internally in a file-based repository or a designated folder system within the operating system. The execution data are stored as lists of parameter values for a given tool, and they can appear in the form of files, especially in the case of execution results. These lists are currently made of value sets per person. Two different values for a given parameter can refer to the same person if the parameter refers to a biological marker measured more than once over time (eg, a person's white blood cell count).

Graphical user interfaces. Two basic functionalities are available to the end users, via a graphical environment. The first involves editing the contents of databases. Using the MVC models as blueprints, suitable views can be automatically created per table, containing data fields that match the table columns. Therefore, the user can perform all create, read, update, and delete (CRUD) operations on the stored objects. Input data are validated before committing a transaction according to pertinent information already located in the databases or defined in the model implementations (eg, preventing the user from setting a biological parameter value out of its permitted range, which is stored in the parameter table).

As it pertains to tool executions, in the simplest of cases, the user can be provided with a view for a particular tool that can present its basic information, a list of the input parameters, and prompt the user to complete them and start the execution. The input fields for these parameters can be either explicitly included in the view if it is preconstructed with handwritten code or received dynamically from the database with a query to the parameter table asking for the input parameters of the tool. This way the form is created on-the-fly thus being always automatically updated with the latest changes of the tool. The view is responsible for presenting the results of an execution to the user. It also provides the ability to produce printable reports (eg, in PDF files). For multiple tools directed toward the same disease or addressing different aspects and variations of a given disease (eg, showing all tools addressing one specific type of cancer), the view can take the form of a wizard, which enables the user to initially determine the disease or variation they want to experiment with and then choose the desired tool from a suitably selected set. To further distinguish this tool execution functionality from the rest of the infrastructure GUIs, this wizard-based environment will be referred to as the *tool execution module*.

API. This is the infrastructure gateway with respect to remote connections and data exchanges with third-party applications. The representational state transfer (REST) architectural style is used, and the exchanged data are in JavaScript object notation (JSON) format. Similar to the previous component, database CRUD capabilities are offered, albeit on a larger scale (actions on multiple or all entries of a table or on a query-determined set of them), by using all the classic HTTP methods (GET, POST, PUT, DELETE, PATCH). The set of the included web services is accessible by a URL system, the form of which is determined through the API implementation software. For simplicity, a common form is suggested with the URL system of the controllers.

Security. The infrastructure is accessed from two different types of users (humans and third-party applications). This dictates the required security measures that provide authentication and authorization services. For physical users, a username and password system is utilized, which also supports granting specific access rights to certain parts of the infrastructure to distinct user group profiles (eg, clinician, modeler, etc.). For applications requiring access to the infrastructure, appropriate web protocols are used. A common solution is OAuth2.0,[8] which is used by companies like Google and Facebook. External security mechanisms can also be used in cases where the infrastructure is part of a larger platform (eg, a trusted third party can handle all the identification and the chosen user group rights policies and provide access via a single sign-on mechanism). In this case, all that is required is the notification/alteration of the controllers to allow the

requested functionality and communicate with the established mechanism. For a standalone version, the security features are included as parts of the infrastructure. Data about individual users, user groups, and access rights can be stored in tables in the repository databases (either integrated in the data repository or as a separate database). The login and signup procedures are performed through the views. Finally, the API with the proper additions to its pool of web services can undertake the identification of external applications before any remote data exchanges take place.

Individual engines. To enhance the overall operation of the infrastructure, specialized engines are used occasionally, which perform specific processes pertaining to either the transformation of the stored data or performance optimization.

The tool execution engine is tasked with the management of all requested and pending tool executions. Using the trifecta of a message broker, a task/job queue processing mechanism and an engine-dedicated result backend database, any requested tool execution is treated as a separate task and coded properly, while its status is monitored. The relevant information is stored in real time in the result backend database and can also be sent to any tables-logs in the repository databases. The need for such a component emerges implicitly from the diversity of the stored tools since, depending on the disease, the biological level whose functions are simulated and the methods of development used for the tool creation, it is possible that tool execution times can vary, reaching from a few seconds up to hours.

The metadata engine can produce Resource Description Framework (RDF) graphs from the contents of the repository databases. Given a set of ontologies and a suitable mapping schema, sets of RDF triples are produced from the database tables relying on the fact that each cell of a table can be mapped to an RDF statement.[9] The output can be sent for storage to external RDF triple stores and by extension published through SPARQL end points.

Implementation Paradigms

The majority of the infrastructure components were implemented using the Python programming language through various software developing tools. More specifically, the Django framework[10] is used for implementing the MVC mechanism of the infrastructure "core", Tastypie[11] was used for the API RESTful web services, and Celery[12] was chosen as a task creation mechanism for the tool execution engine. The engine operates by combining Celery with a message broker (RabbitMQ[13]) and a separate result backend database (MongoDB[14]). Security is handled by the inherent Django user authentication system and the Django OAuth Toolkit,[15] which is used to verify external applications requesting access, using the OAuth2.0 standard. Finally, the relational databases are built using MySQL.

It should be noted that according to the Django interpretation of the MVC pattern, a *view* is used as a callback function for a particular URL. This means that, in an attempt to separate content from presentation, a view "determines which data are presented to the user, not how they are presented".[10] Data presentation is handled by the so-called *templates*, which, in the majority of cases, are implemented as html pages that are sent to the user's screen. Therefore, a more accurate description of the approach would be *model–template–view* (MTV). This means that in the previous sections of component description, the MVC *controllers* are the Django *views* and the MVC *views* are the Django *templates*. From this point onward, the MTV naming convention will be used.

The nephroblastoma paradigm. Based on the blueprint of Figure 2, a working prototype application, named IAPETUS has been implemented in the context of the MyHealthAvatar project.[16] The initial objective was to create a repository that would host the models to be developed and used for the project purposes, along with the necessary execution data. Following the principles of the MVC – or in this case, MVT – pattern, it was possible to capitalize on the potential for scalability offered by Django and include the tool execution module.

The primary example is the nephroblastoma (Wilms tumor) Oncosimulator[11,12] developed by the *In Silico* Oncology and *In Silico* Medicine Group, Institute of Communication and Computer Systems, National Technical University of Athens, the execution results of which, in conjunction with the infrastructure implementation, constitute one of the four high-end clinical use cases of project. The final product is demonstrated and evaluated through the creation of specialized workflows for the two distinct user roles (modeler, clinician) and their respective outputs.

The nephroblastoma (Wilms tumor) Oncosimulator. The nephroblastoma Oncosimulator is a model that simulates tumor response to preoperative chemotherapy treatment with actinomycin and vincristine. It is developed with discrete mathematics, following a top–down approach. The model starts from the macroscopic high biocomplexity level (imaging data) and proceeds toward lower biocomplexity levels. The macroscopic anatomic region of interest is either manually or semiautomatically annotated by the clinicians on MRI imaging sets acquired at the time of diagnosis. A virtual cubic mesh is used for the discretization of the area of interest (tumor) of which the elementary cube is termed geometrical cell. A hypermatrix, ie, a mathematical matrix of (matrices of (matrices… of (matrices or vectors or scalars))), corresponding to the anatomic region of interest is subsequently defined. The latter describes explicitly or implicitly the local biological, physical, and chemical dynamics of the region.[4,17,18]

The basic mechanism of the model is based on the separation of biological cells within a geometric cell (otherwise known as volume element or voxel) through hypermatrix into equivalence classes. Each cell, depending on its mitotic potential, and its current phase of the cell cycle, is assigned to one of these classes. Then, a series of status variables (oxygen

and glucose concentration, cell number in the voxel and number of cells hit by therapy), and the application of transitional algorithms from a cellular state to another in periodical time intervals, determine the next overall state of the voxel and its cells.

The nephroblastoma Oncosimulator is implemented using the C++ language. The primary outputs of the model are a series of RAW image files displaying the tumor volume for each day of the treatment period and a set of DAT files containing all the numerical results (tumor growth percentage, etc.).

Workflows. The MyHealthAvatar project is a feasibility study,[16] aiming to collecting personal data and utilizing them to advance healthier lifestyles through various processes, including disease prevention. This has led to a scenario-based design, including use cases and workflows that implement them.[19]

The nephroblastoma use case addresses directly two out of the four different categories of users defined in MyHealthAvatar, modelers (also corresponding to the more general category of biomedical or basic science researcher) and clinicians.[19] Modelers are expected to focus mostly on the functionality of repositories, while their calls to the tool execution module will generally be limited in number and their sole purpose would be to further calibrate and fine-tune their creations. On the other hand, doctors are expected to ask for a lot more tool executions. As such, any involvement with the storage facilities and/or the API will be limited to uploading and retrieving medical data and saving result reports. With the exception of

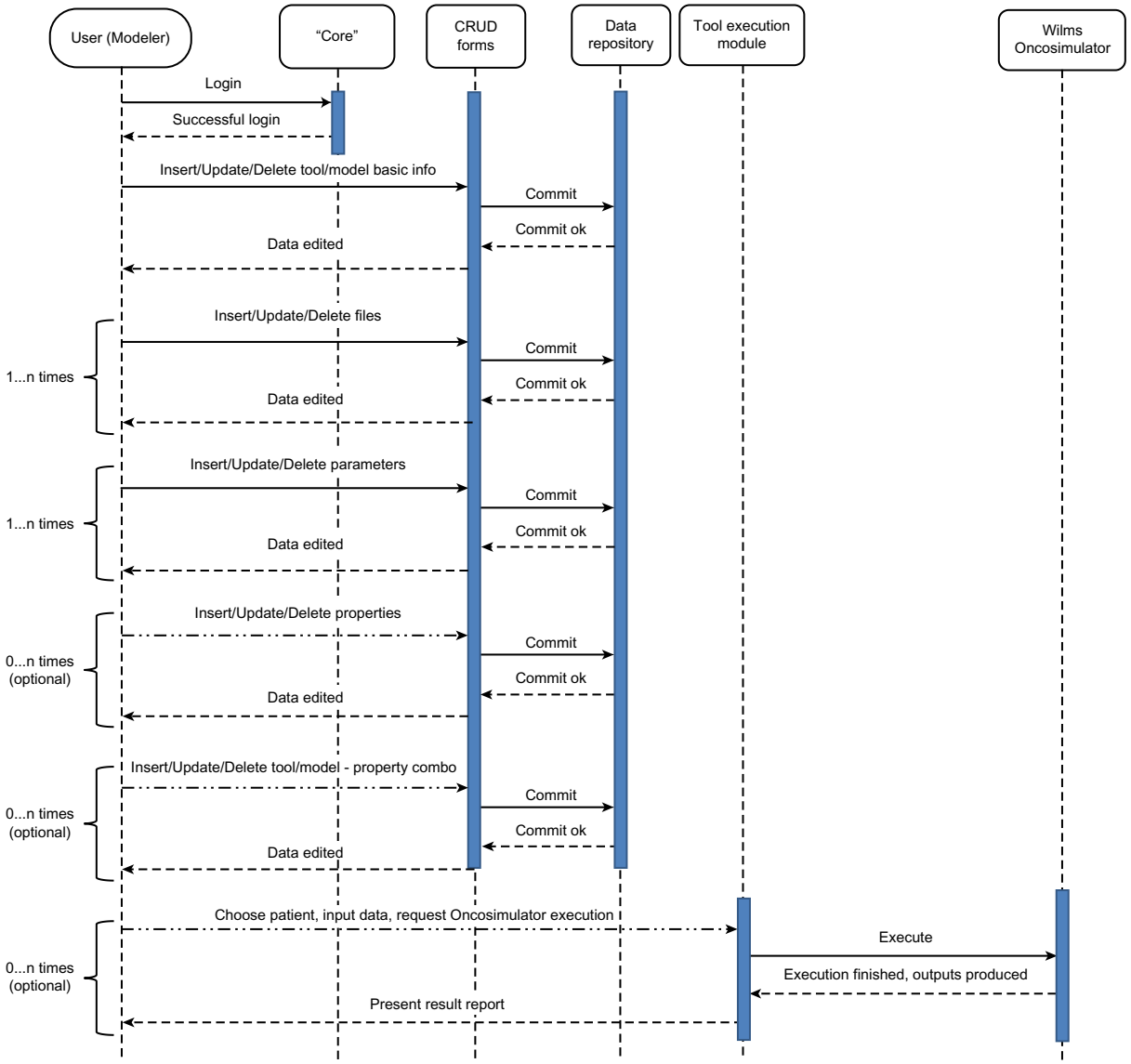


Figure 3. Proposed workflow for a modeler using the infrastructure. To register a tool/model in the repository data must be entered for at least the tool/model's basic information, files, and parameters. Additional optional choices are the addition of parameters and/or the tool/model's association with other existing properties. Afterward, the modeler has the option to conduct test runs with the execution module. Required user actions for the workflow are depicted using solid right arrows, whereas optional actions are depicted with double-dot and dash right arrows.

medical data uploading, all other pertinent commands will be given through the tool execution module, to ensure minimal direct involvement with the repository CRUD mechanisms.

The modeler workflow is shown in Figure 3. After logging to the system, they choose to enter the tool and model repository. Assuming that the objective is to upload a simulation model for storage, data must be entered in at least three tables. First the general information table, where the name and description of the tool will be given, followed by the uploading of all tool pertinent files to the file table (executable file, documentation, auxiliary scripts for additional functions such as visualization of results). Finally, a list of input and output parameters should be defined, which will facilitate the construction of the input form and the result report from the tool execution module.[20,21] Defining properties and relating them to the uploaded tool is encouraged, albeit not compulsory.

Respectively, the clinician, after logging, would enter the repository only to register information of a new patient and/or to provide one or more sets of values for a certain patient, in order to use them subsequently as input for a simulation model. Then, they proceed to the tool execution module forms. The interface wizard will prompt them to choose the following: disease, model to use, and patient name, in that order. Then, a form appears with the appropriate input fields derived from the rows of the parameter storage table, which pertain to the chosen simulation model. After providing the desired inputs, the model performs an execution. The results are displayed on the last wizard screen and can be saved as a PDF file. Similarly, at a future point in time, the clinician can download the saved report through the interface of this application.[20] The workflow is shown in Figure 4.

Legal/ethics/security considerations. The development of the IAPETUS prototype as part of the MyHealthAvatar project was subject to a set of legal and ethical guidelines. These guidelines stemmed from the notion that the citizen becomes the main stakeholder and is able to freely upload and use their own data in conjunction with any and all available software tools developed in the project context. Especially in the case of patient-centered simulation models, privacy and data protection issues could arise. Prior to the use of data from a simulation model, the data owner should be aware of how the model will use their data, what new data will emerge from the simulation results and who will be the owner of these data. A simulation model should also operate under a security framework that prevents data loss or usage by unauthorized parties. Furthermore, the interpretation of execution results can lead to liability issues. A citizen is not necessarily expected to know the exact clinical meaning of all the data produced by a simulation model execution. Therefore, any attempts by the citizen to evaluate their own medical condition, outside of a clinical environment, could lead to misinterpretations and thereby negatively affect the person's psychological state.

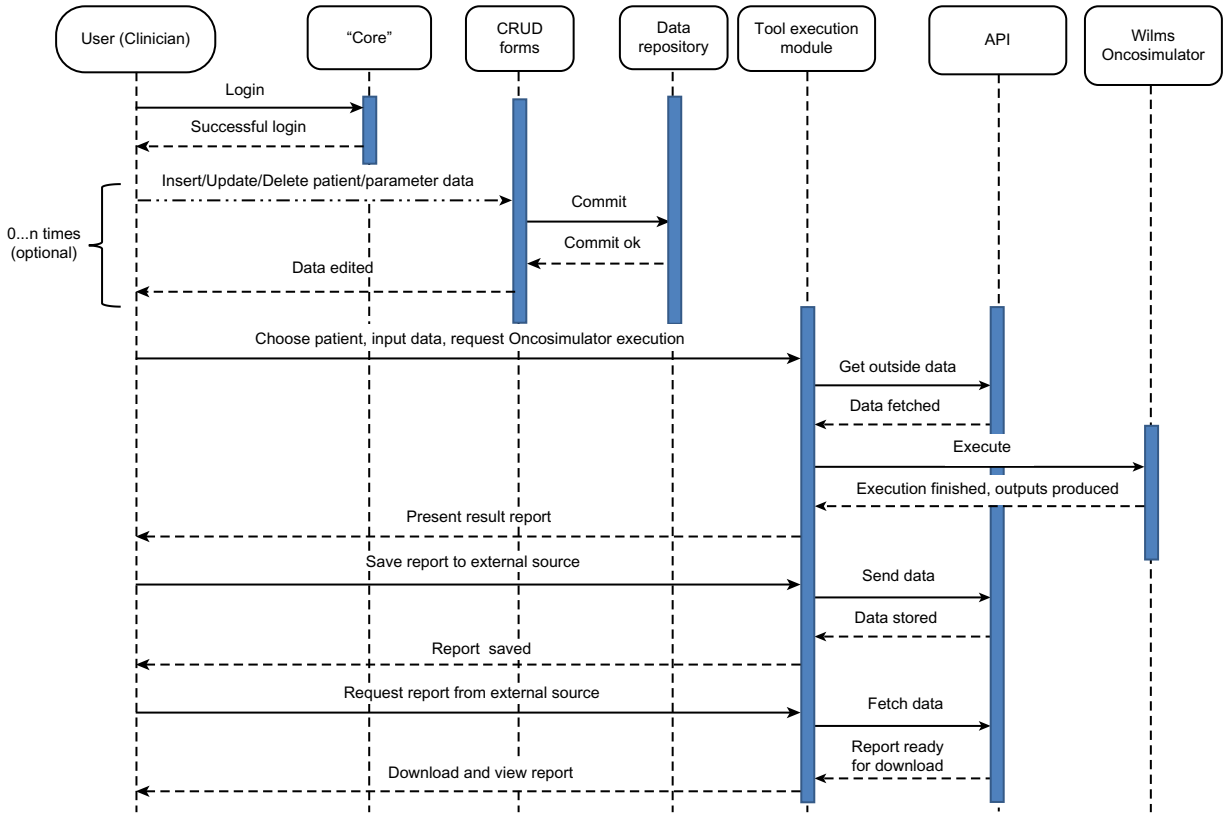


Figure 4. The proposed workflow for the clinician. Required user actions for the workflow are depicted using solid right arrows, whereas optional actions are depicted with double-dot and dash right arrows.

According to the doctrine of informed consent,[22] the citizen must be informed about these issues before giving his/her consent for the use of his/her personal data.[23]

The aforementioned issues, in conjunction with the types of stakeholders to whom the present work is addressed, dictate the choice and implementation of the IAPETUS security mechanisms. In order for a citizen's data to be used, a key assumption is made: a clinician can use personal data, only under the specific informed consent of the data owner – in this case, the citizen – and on condition that both parties are interacting within the confines of a clinical environment. This means that the clinician and the citizen should be in the clinician's office, at the same time, in front of the same device (personal computer, laptop, tablet, etc.). The use of the citizen's data as input for one or more simulation models is allowed only if the citizen has previously given his/her permission for such actions to be taken and after an initial proper briefing regarding these actions has taken place. If the infrastructure is hosted in a large healthcare provider (eg, a hospital), then a clinician can view, edit, and use the data that pertain only to the patients, whose consent they have already obtained. Similarly, to protect the modelers' intellectual property rights regarding the models they have created, a tool and its related information (files, parameters, and property values) can be viewed by all IAPETUS users, but altered only by their creator. The creator can also determine which of the files that they uploaded to the infrastructure can be downloaded by other users and which files cannot.

In Django, all permissions are handled by the inherent authentication and authorization system. By explicitly specifying the user roles of modeler and clinician and defining the corresponding role rights, it is possible to prevent modelers from performing CRUD operations on personal patient data and clinicians from altering tool parts and specifications. Furthermore, the schema of the relational databases can be updated by placing additional fields to store the creator of an object, where necessary. Therefore, when a user views the object information, the comparison of the creator and viewer identities will enable or disable the editability of the returned template. In the case of machine-to-machine communication, the HTTPS protocol is used for the IAPETUS URL system, through the use of Python's Django-sslserver[24] package. Finally, if the users access the infrastructure from devices using static IP address settings, then Django offers the option to allow access only to specific IP addresses.

The breast cancer paradigm. This paradigm is based on another simulation model, the breast cancer Oncosimulator, which has been developed outside the scope of the MyHealthAvatar project. It aims to demonstrate the versatility of the infrastructure by highlighting the general principles that were used to create the infrastructure database schema along with the workflows developed for the two different kinds of users and how they can be used to accommodate any kind of tool.

Furthermore, the model is developed using MATLAB, a language which, in contrast to lower level programming languages, such as C++, is very tightly coupled to its own execution environment, which is the MATLAB Compiler Runtime (MCR).[25] This arrangement helps in demonstrating the ability of the infrastructure to address the end user's predicament of installing third-party software in their devices to access and use the tool and model repository contents. All stored tools reside in the same physical machine as the infrastructure. In addition, the same device is used for tool executions and result storage. Therefore, any and all software that is required for the tool executions should be installed only in the infrastructure deployment machine. This allows access to the infrastructure using devices such as smartphones and tablets, which have less computational power in comparison to personal computers and laptops and therefore are less probable to have highly specialized software such as MCR installed within them.

The breast cancer Oncosimulator. The breast cancer Oncosimulator simulates the vascular tumor growth and the response to antiangiogenic treatment of breast cancer, through the administration of bevacizumab, a monoclonic antibody that prevents the connection of the vascular endothelial growth factor (VEGF) with the corresponding receptors on the endothelial cells surrounding the tumor. It is based on a system of ordinary differential equations, which describe the tumor volume and its carrying capacity, ie, the maximum tumor volume that can be supported by the given vasculature.[26,27]

The model is implemented as a set of MATLAB M files. One of them assumes the role of the *master* script and subsequently calls the others. The input parameters are fed as a Comma-Separated Values (CSV) file and the output is a diagram in which the tumor volume and its carrying capacity over time are plotted.

Results

Tool and model repository outputs. The repository final schema is demonstrated in Figures 5 and 6 for the nephroblastoma paradigm and in Figures 7 and 8 for the breast cancer paradigm. These figures are practically the outputs of the modeler workflow, which is common for both paradigms. They display the grouping and separation of tool components and characteristics, as well as the data input procedure. This method achieves universal support for every simulation model, regardless of the simulated disease, the implementation, and/or any number of auxiliary tools that may be required for its operation. Figure 9 illustrates this feature, by providing a full list with the tools and models, which are currently stored in IAPETUS.

Finally, to demonstrate the functionality from the perspective of machine-to-machine communication, Figure 10 displays the information about the tools from Figure 9, in JSON format, as the response of a remote call to the IAPETUS API. For purely esthetical reasons, the returned

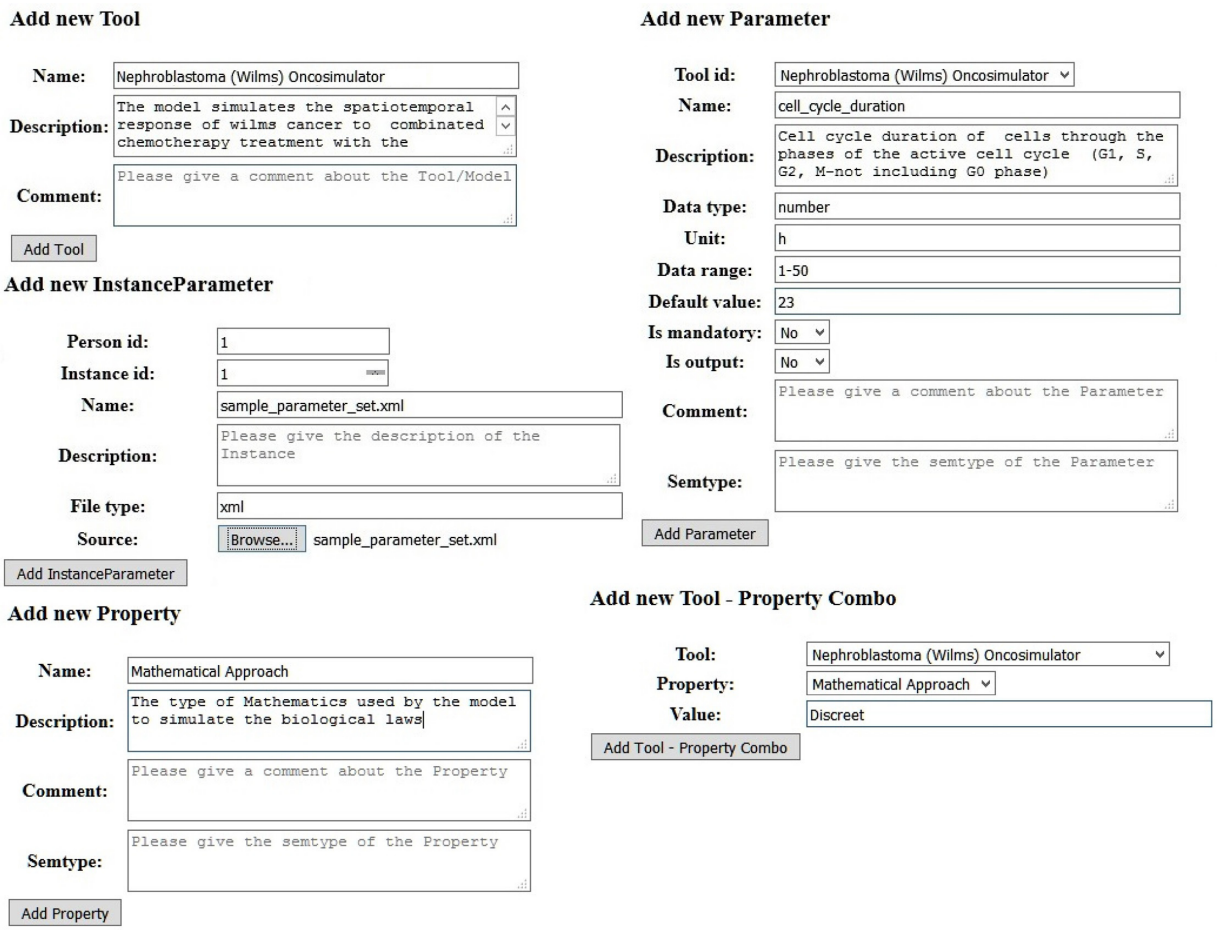


Figure 5. Consolidation of forms for the basic database tables with inputs for the nephroblastoma paradigm.

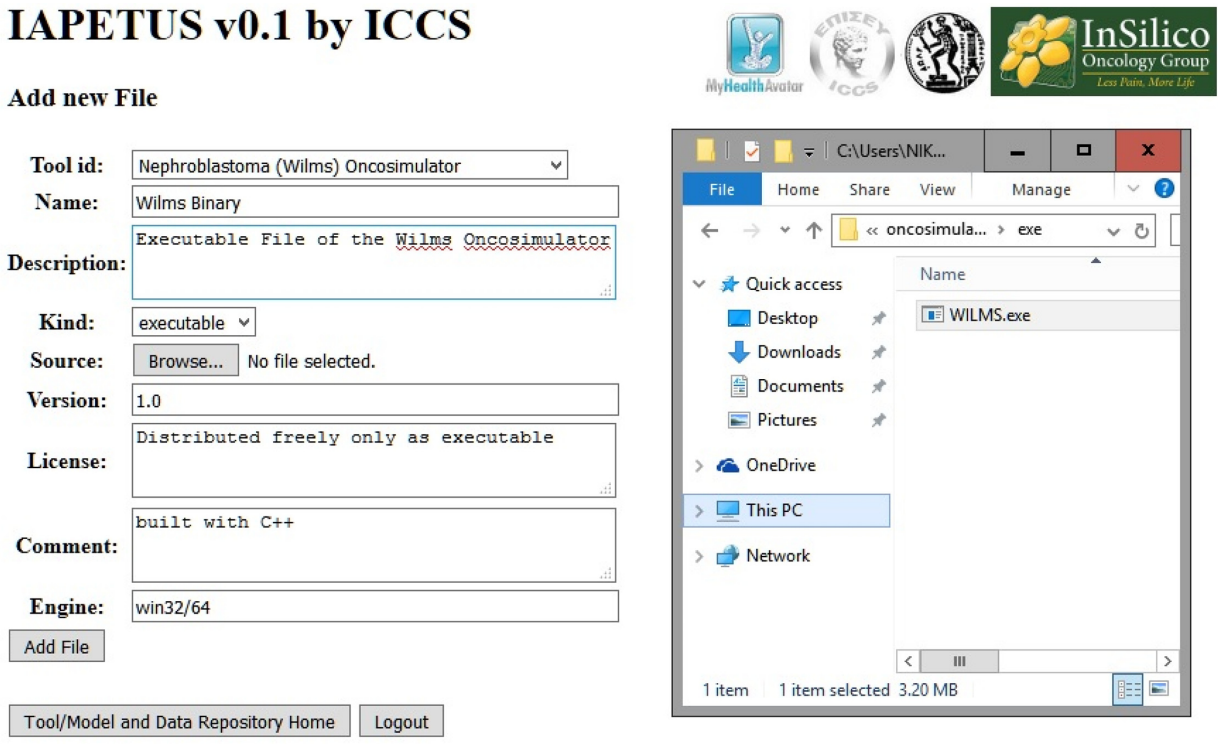


Figure 6. Nephroblastoma paradigm – file uploading form with input – searching for the file to upload.

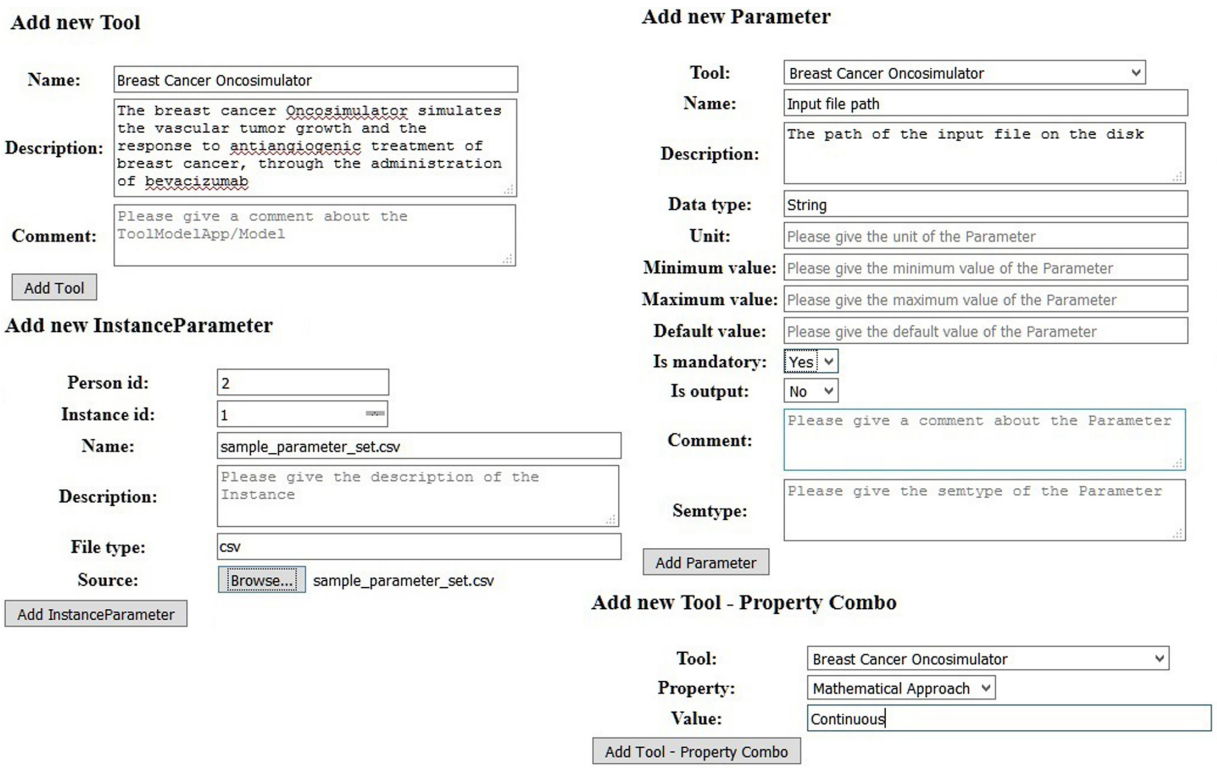


Figure 7. Consolidation of forms for the basic database tables with inputs for the breast cancer paradigm.

JSON content is sent through a validator to make the proper indentations.

Model execution. The nephroblastoma Oncosimulator execution results associated with the clinician's workflow are given in Figure 11. Since result evaluation in the context of clinical practice is the end goal, converting them into clinically comprehensible forms is required. Therefore, an extra script file written for Gnuplot[28] was added to the nephroblastoma Oncosimulator related files. A new tool was also created and its implementation as a MATLAB compiled executable file was added. The Gnuplot script produces a graph of the change in total tumor volume over time. The new MATLAB tool produces an image file displaying the superposition of the initial and final state images of the tumor, which are reconstructed from the RAW files of the first and last day of the treatment scheme. Figure 12 demonstrates the respective results for the breast cancer Oncosimulator, which consist of a single diagram showing the tumor volume and carrying capacity graphs.

Connection with external data sources. In order to demonstrate the capabilities of IAPETUS to cooperate with external data sources and exchange data as needed, the nephroblastoma high-end use case, as defined in MyHealthAvatar,[29,30] required the establishment of a collaboration with the Computational Horizons in Cancer (CHIC) project.[5] The nephroblastoma Oncosimulator needs preprocessed imaging data to operate.[17,18] To meet this demand, IAPETUS was linked to a clinical data repository developed to hold the data used in CHIC,[31] which acts as one of the third-party sources of data. Using a remote call to the CHIC repository API, IAPETUS receives a pair of a RAW image file and its matching MHD header file. These files are considered to be the personal data of a hypothetical synthetic patient, due to them being actually pseudonymized, in order to comply with the legal and ethical frameworks of both projects. These files can indeed be stored in IAPETUS data repository, since the pertinent schema allows the storage of both real and pseudonymized patients. Figure 13 shows how the incoming data are presented to the end user in the tool execution wizard data input form. Finally, the model is executed with the selected inputs. The results are displayed on the last wizard screen and can be saved as a PDF file. Again, for the purposes of MyHealthAvatar, this PDF file is uploaded to the central platform produced by the project by making a call to the platform API. Similarly, at a future point in time, the clinician can download the saved report from the central platform through the interface of IAPETUS.

Performance measurements. Given the fact that the average patient visit time in a oncologist's office can reach up to 23 minutes,[32,33] and it can be reduced down to an average of 10 minutes for other specialties,[34] emphasis has been given on reducing the time required to get clinically relevant results from IAPETUS, focusing on two basic points: the use of the tool execution engine for handling multiple execution requests and producing user-friendly graphical interfaces, based on wizards, to quickly guide the clinician through the data input and output evaluation procedures. To that end, time measurements have been taken for both paradigms, featuring the clinician workflow in the cases of a single execution, two and three

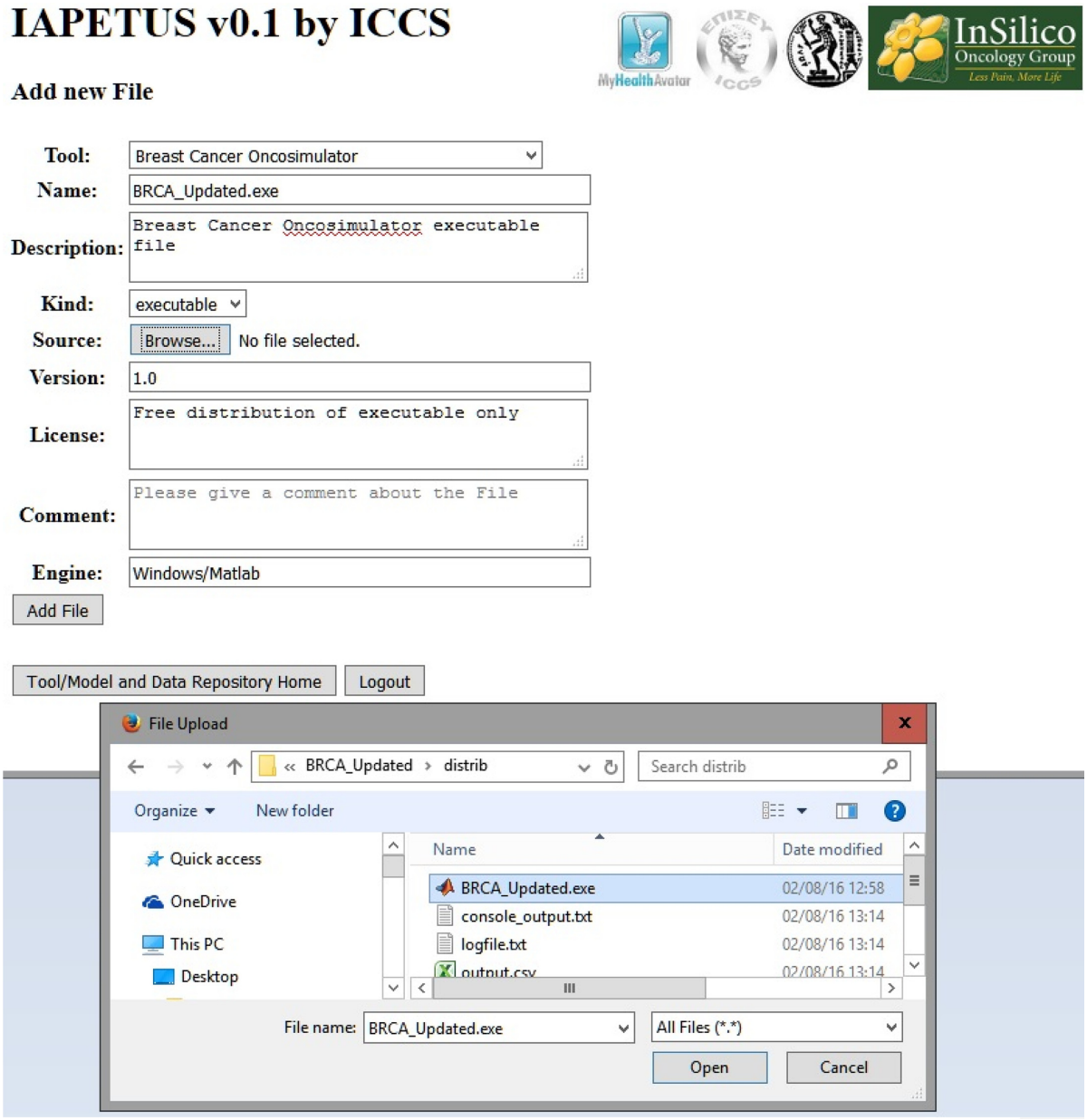
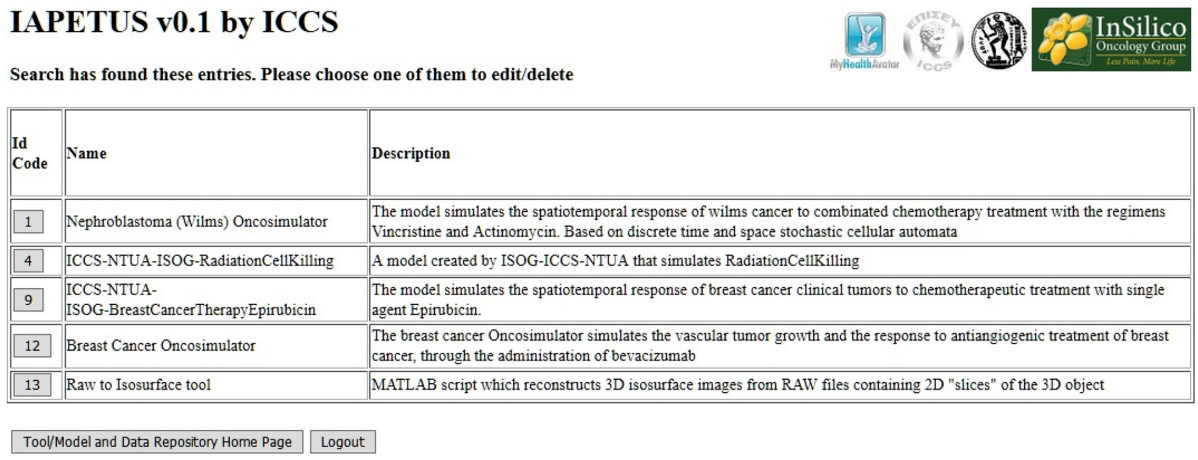


Figure 8. Breast cancer paradigm – file uploading form with input – searching for the file to upload.



Id Code	Name	Description
1	Nephroblastoma (Wilms) Oncosimulator	The model simulates the spatiotemporal response of wilms cancer to combinated chemotherapy treatment with the regimens Vincristine and Actinomycin. Based on discrete time and space stochastic cellular automata
4	ICCS-NTUA-ISOG-RadiationCellKilling	A model created by ISOG-ICCS-NTUA that simulates RadiationCellKilling
9	ICCS-NTUA-ISOG-BreastCancerTherapyEpirubicin	The model simulates the spatiotemporal response of breast cancer clinical tumors to chemotherapeutic treatment with single agent Epirubicin.
12	Breast Cancer Oncosimulator	The breast cancer Oncosimulator simulates the vascular tumor growth and the response to antiangiogenic treatment of breast cancer, through the administration of bevacizumab
13	Raw to Isosurface tool	MATLAB script which reconstructs 3D isosurface images from RAW files containing 2D "slices" of the 3D object

Figure 9. List of stored tools and models. In addition to the models pertaining to the two paradigms, a universal tool developed in MATLAB is stored, which produces three-dimensional reconstructed volume images from RAW files containing sets of two-dimensional *slices* of the volume.

```
{
    "meta":{
        "limit":20,
        "next":null,
        "offset":0,
        "previous":null,
        "total_count":5
    },
    "objects":[
        {
            "comment":"",
            "description":"The model simulates the spatiotemporal response of wilms cancer to
            combinated chemotherapy treatment with the regimens Vincristine and Actinomycin.
            Based on discrete time and space stochastic cellular automata",
            "id":1,
            "name":"Nephroblastoma (Wilms) Oncosimulator",
            "resource_uri":"/repository/api/tool/1/"
        },
        {
            "comment":"my comments",
            "description":"A model created by ISOG-ICCS-NTUA that simulates RadiationCellKilling",
            "id":4,
            "name":"ICCS-NTUA-ISOG-RadiationCellKilling",
            "resource_uri":"/repository/api/tool/4/"
        },
        {
            "comment":"",
            "description":"The model simulates the spatiotemporal response of breast cancer
            clinical tumors to chemotherapeutic treatment with single agent Epirubicin. ",
            "id":9,
            "name":"ICCS-NTUA-ISOG-BreastCancerTherapyEpirubicin",
            "resource_uri":"/repository/api/tool/9/"
        },
        {
            "comment":"",
            "description":"The breast cancer Oncosimulator simulates the vascular tumor growth
            and the response to antiangiogenic treatment of breast cancer, through the
            administration of bevacizumab",
            "id":12,
            "name":"Breast Cancer Oncosimulator",
            "resource_uri":"/repository/api/tool/12/"
        },
        {
            "comment":"",
            "description":"MATLAB script which reconstructs 3D isosurface images from RAW files
            containing 2D \"slices\" of the 3D object",
            "id":13,
            "name":"Raw to Isosurface tool",
            "resource_uri":"/repository/api/tool/13/"
        }
    ]
}
```


Figure 10. JSON representation of the data in Figure 9.

simultaneous Oncosimulator executions with and without the use of the execution engine.

For the nephroblastoma paradigm, the treatment schema given as input to the Oncosimulator for all tests was the same: four administrations of vincristine, one every 7 days, and two administrations of actinomycin, one every 14 days. The administrations start on the seventh day after diagnosis, and posttreatment surgery takes place one day after the final vincristine session. The breast cancer paradigm simulates a treatment scheme based solely on bevacizumab. The simulated schema is composed of nine treatment sessions. In each session, 10 mg/kg of drug is administered. Sessions take place twice a week. This means that if the first administration takes place on day one, then the next sessions are determined by adding three or four days alternately.

For each case, the application ran 10 times and the average value was calculated. The tests were conducted on a PC with an Intel Core-i7 3.4 GHz CPU with 16 GB RAM. The results are presented in Table 1 for the nephroblastoma Oncosimulator and in Table 2 for the breast cancer Oncosimulator.

By examining Table 1, it can be deducted that the engine helps the infrastructure to produce results 48% and 56% faster for two and three simultaneous executions of the nephroblastoma Oncosimulator, respectively. The corresponding numbers for breast cancer Oncosimulator are 58% and 71%. In addition, from the numbers of the third row, it can easily be concluded that without the execution engine, multiple calls to run the nephroblastoma model are processed in a serial way, whereas the calls to the breast cancer model are processed even slower.

Furthermore, comparing these results, and especially the overall execution time for the clinician workflow with the average patient visit time, leads to the conclusion that the infrastructure can play a crucial role in aiding the clinician's work. Low deliverance times means more executions per session, or more time for cross-examining the execution results combined with data related to drug allergies, comorbidities, etc., thus contributing to the notion of personalized medicine by allowing a greater portion of these data to be factored in the final decision of the proper

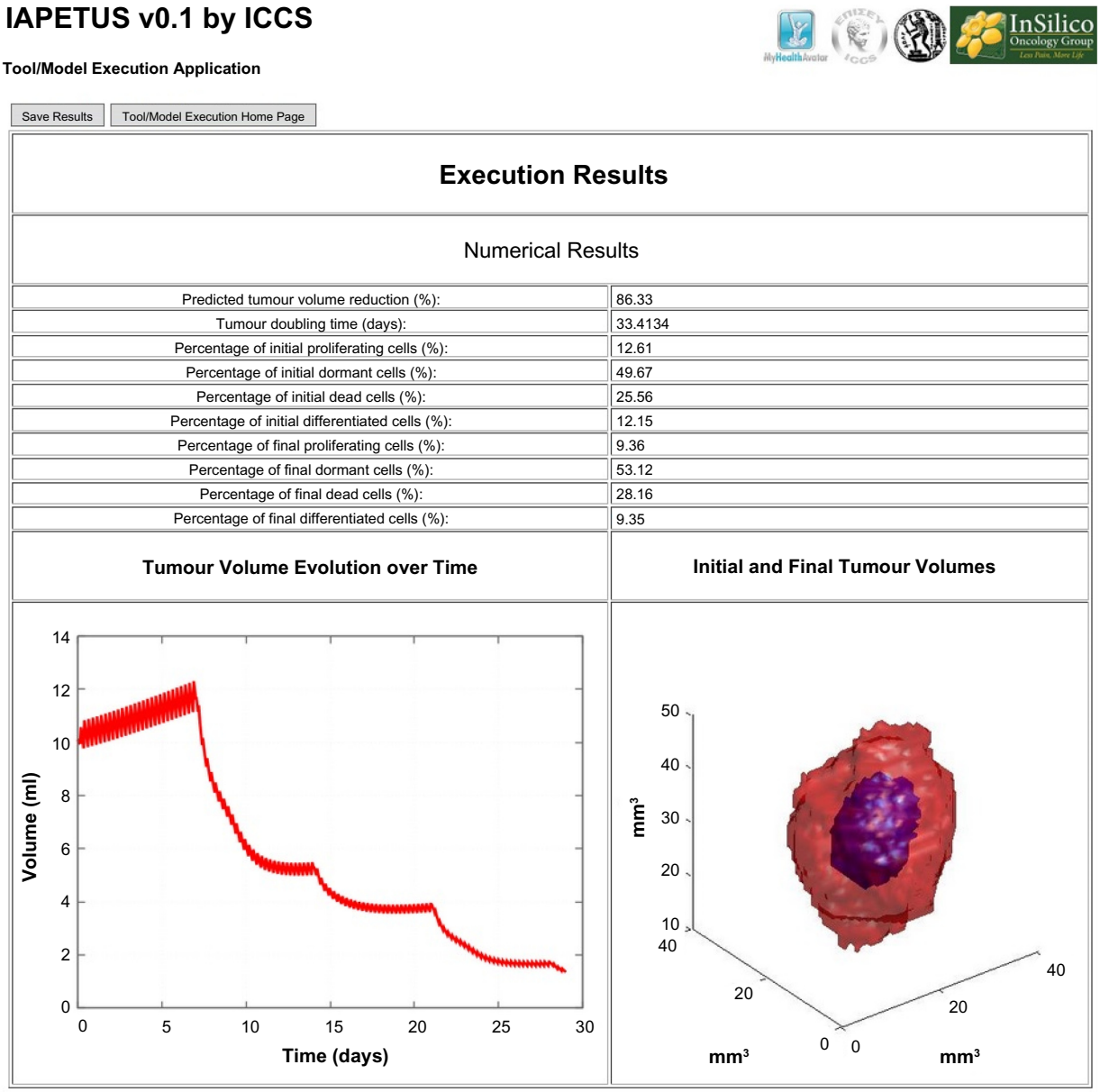


Figure 11. The results from the nephroblastoma Oncosimulator execution. In the upper part, numerical values of medical importance are presented, taken from the produced DAT files. From the same source, the *Tumor Evolution over Time* graph (lower left) is plotted. Finally, on the lower right side, the image or superimposed initial and final tumor images are taken from the produced RAW files.

treatment strategy. With the help of the engine and proper alterations in the interfaces, the doctor can just as easily compare 2, 3, or more different treatment schemas in an expected slightly higher time (which can be predicted if we add to the overall clinician workflow time, the difference between the second or third and the first value of the second row for 2 or 3 schemas, respectively), under 90 seconds, nonetheless.

Infrastructure evaluation. According to the MyHealthAvatar description of work, all developed software, workflows, and use cases were subjected for evaluation. To that end, a series of workshops and sessions were conducted[21,35] and the evaluators, after the necessary demonstration of the functionalities, were requested to provide some feedback. To that end, a number of online questionnaires were created. The questionnaires were composed of distinct sets of questions pertaining to topics such as functionality, maintainability, portability, and quality of use. Each question could be answered in a numerical Yes–No scale from 1 to 5, with 5 being the most positive answer and 1 being the most negative.[36]

In the context of the present work, the results of two of these questionnaires will be examined. These questionnaires refer to the tool and model repository, which were demonstrated through the modeler workflow and the tool execution module, which was named *clinical scenario* in the project and was demonstrated via the clinician workflow. The nephroblastoma Oncosimulator was used as a primary example to demonstrate the workflows and was included in a number of

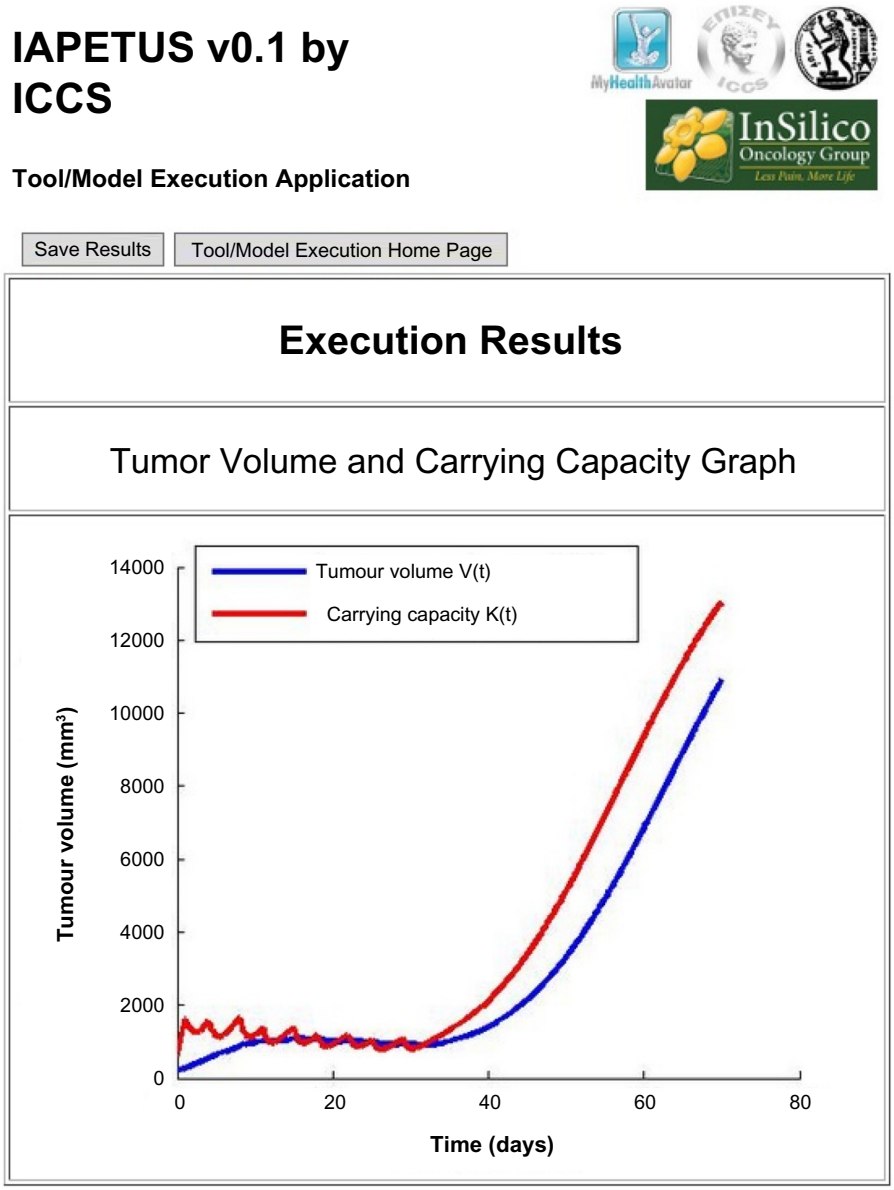


Figure 12. The results from the breast cancer Oncosimulator execution. The graph is automatically produced from the MATLAB executable file.

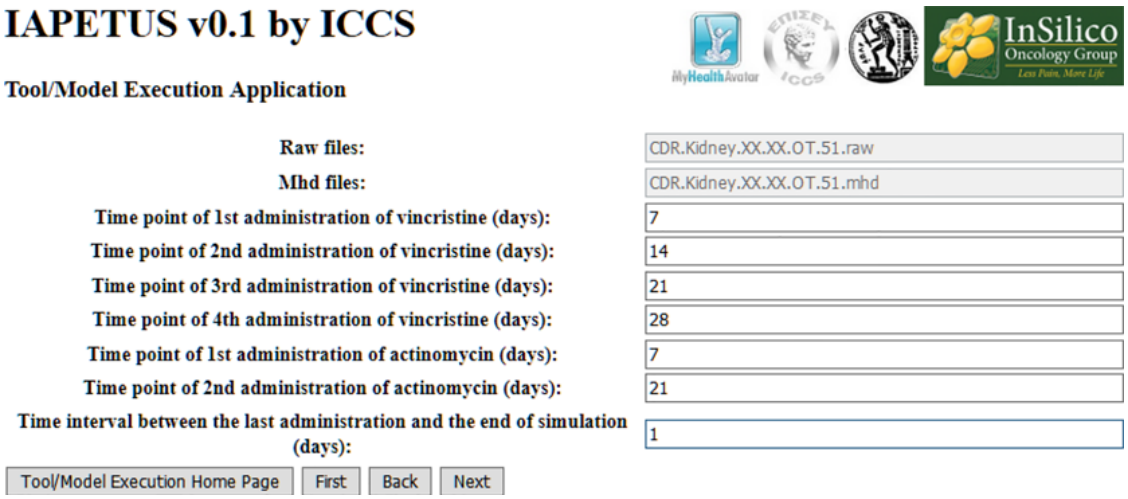


Figure 13. The tool execution wizard data input form for the nephroblastoma high-end use case. Above the fields that accept the treatment schema, the image-header file pair is displayed. These files are considered to be the personal data of the patient that were selected in the wizard's previous step.

Table 1. Execution time measurements of the nephroblastoma Oncosimulator.

NUMBER OF EXECUTIONS / EXECUTION MODE	SINGLE EXECUTION	2 SIMULTANEOUS EXECUTIONS	3 SIMULTANEOUS EXECUTIONS
Command prompt only	19.12s	20.43s	21.81s
Using the infrastructure (with execution engine)	25.67s	28.05s	35.64s
Using the infrastructure (without execution engine)	26.82s	53.62s	1 min 20.88s
Clinician workflow overall time (login to report retrieval, single execution only)	1 min 16.74s	1 min 19.12s (estimated)	1 min 26.71s (estimated)

Table 2. Execution time measurements of the breast cancer Oncosimulator.

NUMBER OF EXECUTIONS / EXECUTION MODE	SINGLE EXECUTION	2 SIMULTANEOUS EXECUTIONS	3 SIMULTANEOUS EXECUTIONS
Command prompt only	4.74s	5.37s	6.59s
Using the infrastructure (with execution engine)	6.49s	6.17s	6.42s
Using the infrastructure (without execution engine)	7.74s	14.83s	22.29s
Clinician workflow overall time (login to report retrieval, single execution only)	28.85s	39.06s (estimated)	48.51s (estimated)

Table 3. Questionnaire and average results for the modeler workflow evaluation.

QUESTION	MEAN VALUE
Functionality	
Can this web application store models?	4.6
Can this web application store model attributes? (parameters, etc.)	4.4
Can this web application store data to use with the models?	4
Can this web application search and present stored data?	3.7
Can this web application retrieve data in files?	4.1
Can this web application alter its stored data?	4.3
Efficiency	
How quickly does the repository respond to the user requests?	4.6
Is the application comprehensible?	3.2
Is support of a technical person needed in order to use this application?	3.3
Compatibility	
Do you know other similar application? If yes, is this tool better than the other you know?	2.5
Usability	
Can you comprehend the application's functionalities?	3.7
Can you learn to use the application easily?	2.9
Can you use the application without much effort?	3.7
Does the interface look good?	2.3
Does the interface provide all required information?	3.1
Reliability	
Have most of the faults in the software been eliminated over time?	2.6
Is the software capable of handling errors?	3.3
Can the services resume working & restore lost data after failure?	3.3
Portability	
Can the web application be easily accessed from any pc?	5
Security	
Are data accessible only to authorized users?	4.7
Do you think the uploaded data are secure?	3.7

(*Continued*)

Table 3. (*Continued*)

QUESTION	MEAN VALUE
Does the system prevent unauthorized access?	4.3
Maintainability	
Can the software be tested easily?	3.2
Quality in use	
How accurate and complete is the software for the intended use?	3.2
Does the software improve the time or reduce resource for the intended goal?	4
Does the software satisfy the perceived achievement of pragmatic goals?	3.7
Can the software harm people in the intended contexts of use?	1.5

Table 4. Questionnaire and average results for the clinician workflow evaluation.

QUESTION	MEAN VALUE
Functionality	
Can the web application call the Nephroblastoma Oncosimulator?	4.8
Can the web application perform an execution of the Nephroblastoma Oncosimulator successfully?	4.9
Can the application fetch the clinical data (image files) from an outside source (CHIC data repository) successfully?	4.8
Can the user set model input through the application?	4.8
Can the user submit an execution of the Nephroblastoma Oncosimulator model through the web service?	4.8
Is the user interface for execution submission of Oncosimulator user friendly?	4.4
Is the presentation of the results satisfying?	4.1
Is the NEPH – UC clinically relevant?	4.4
Efficiency	
Is the application comprehensible?	4.4
Can you learn how to use the system easily?	4.7
Is support of a technical person needed in order to use this tool?	3
Compatibility	
Is the model running and the results presented independently of the software (windows version/web browsers) available on user's pc?	4.3
Can the system exchange data fluently with external modules?	4.1
Do you know other similar application? If yes, is this tool better than the other you know?	2.4
Usability	
Is the execution of the model easy?	4.9
Is the execution time consuming?	2
Can the tool resume working & restore lost data after failure?	3.9
Does the interface provide all required information?	4.2
Is the produced report useful?	4
Reliability	
How accurate and complete is the software for the intended use?	4
Is the output trustful?	3.9
Are the results presented sufficient for clinical purposes?	4
Portability	
Can the tool be easily accessed from any pc?	4.9
Security	
Do you think your data are secure?	4
Are data accessible only to authorized users?	4.1
Does the system prevent unauthorized access?	4

(*Continued*)

Table 4. (*Continued*)

QUESTION	MEAN VALUE
Quality in use	
How accurate and complete is the software for the intended use?	4.3
Does the software improve the time or reduce resource for the intended goal?	4.7
Does the software satisfy the perceived achievement of pragmatic goals?	4.4
Can the software harm people in the intended contexts of use?	2.1

questions. A grand total of 40 responses were received from modelers and clinicians familiar with simulation models. Tables 3 and 4 list the average answers per question.[36]

The evaluation results produce an overall average of 3.6/5 for the tool and model repository and 4.1/5 for the functionality of the tool execution module, combined with the nephroblastoma Oncosimulator. Both results are above the average scale grade of 3, which indicates that the infrastructure fulfills its basic objectives, especially in the case of tool execution and result presentation and handling. At the same time, the lowest scores dictate the necessary immediate infrastructure updates, which mostly pertain to the enhancement of usability via the improvement of the Django templates presented to the physical users and the transition of the infrastructure to faster computers. A dedicated server or a powerful virtual machine in a cloud infrastructure could provide the necessary computational power to lower the model execution times, which in turn will reduce the overall amount of time needed for the execution of the workflows.

Discussion – Future Work

We have demonstrated the architecture and the prototype implementation of a system for storing and handling the components and related entities of multiscale cancer models in the *in silico* oncology context. Both clinicians and modelers can benefit from the system, as it consolidates all the basic functions that are required for a simulation model to produce results, thus facilitating its translation into clinical practice.

We have adopted multiscale cancer modeling as an excellent paradigm of the broader international virtual physiological human (VPH) initiative. Cancer is manifested at all levels of biocomplexity (from the atomic and molecular up to the whole organism level).[4] It also may dictate the involvement of most medical specialties when it comes to treatment. Our approach is easily extensible to other diseases and conditions since in addition to the previous remarks, both the infrastructure components and the schema chosen for the storage databases regard the tools and models purely as pieces of software, relying on characteristics such as files, textual descriptions, and input/output parameters (the input and output variables that are used by the model source code).

Establishing data exchange channels between the presented infrastructure and repositories and databases of molecular and genomic data available such as ArrayExpress[37] and Progenetics.net[38] could be achieved via an updated set of Django views that would include remote API calls for fast item search and retrieval. Such an addition would allow the feeding of more complex models with various multiscale data.

For the needs of content publishing, searching and enhanced compliance with advanced legal and ethical frameworks, the current infrastructure is intended to be connected with a metadata infrastructure, the blueprint of which has been developed and presented in our previous work.[39] The latter would allow the publishing of the repository contents through SPARQL end points. Finally, as the application is already en route to be redeployed to a private cloud infrastructure, special care will be taken for parallelized models, capable of exploiting multiple CPU cores, starting with an initial version of the Oncosimulator.[40]

Conclusion

In this paper, we have presented a proposal pertaining to the architecture of a modular system based on a tool and model repository, whose objective is to store disease simulation models and supplementary software tools, provide support for their executions, and manage the produced results via an integrated interface based on wizards. Two demonstration paradigms that have served as the basis of the infrastructure design have been considered. These include the nephroblastoma (Wilms tumor) and the breast cancer Oncosimulators. A storage approach allocating the individual model characteristics has been presented. In addition, through time measurements of Oncosimulator executions under various scenarios, the importance of dynamic, ad hoc interfaces has been highlighted. Furthermore, the benefit of exploiting all the available CPU cores of a system in order to reduce the overall time that a clinician should spend on the computer screen in making decisions regarding the treatment strategy for a given patient has been illustrated. Finally, possible future extensions based on the aforementioned notion in combination with the flexibility offered by the selected database schema capable to accept practically any model, regardless of the simulated disease, have been outlined.

Acknowledgments

The authors acknowledge Professor Norbert Graf, University Hospital of Saarland, Professor Feng Dong, University of Bedfordshire, and Dr. Dimitra Dionysiou, Institute of Communication and Computer Systems, National Technical University of Athens, for their support.

Author Contributions

Conceived and designed the experiments: NAC, NET, ECG, KDA, FDM, GSS. Analyzed the data: NAC, NET, ECG, KDA, GSS. Wrote the first draft of the manuscript: NAC. Contributed to the writing of the manuscript: NET, ECG, KDA, GSS. Agreed with manuscript results and conclusions: NAC, NET, ECG, KDA, FDM, GSS. Jointly developed the structure and arguments for the paper: NAC, NET, ECG, KDA, GSS. Made critical revisions and approved the final version: GSS. All the authors reviewed and approved the final manuscript.

REFERENCES

1. Hamburg MA, Collins FS. The path to personalized medicine. *N Engl J Med*. 2010;363(4):301–4. doi: 10.1056/nejmp1006304.
2. Rossi S, Christ-Neumann M-L, Rüping S, et al. p-Medicine: from data sharing and integration via VPH models to personalized medicine. *Ecancermedicalscience*. 2011;5:218. doi: 10.3332/ecancer.2011.218.
3. Viceconti M, Clapworthy G, Jan S. The virtual physiological human – a European initiative for in silico human modelling. *J Physiol Sci*. 2008;58(7):441–6. doi: 10.2170/physiolsci.rp009908.
4. Stamatakos G, Dionysiou D, Lunzer A, et al. The technologically integrated oncosimulator: combining multiscale cancer modeling with information technology in the in silico oncology context. *IEEE J Biomed Health Inform*. 2014;18(3):840–54. doi: 10.1109/jbhi.2013.2284276.
5. Stamatakos G, Dionysiou D, Misichroni F, et al. Computational horizons in cancer (CHIC): developing meta- and hyper-multiscale models and repositories for in silico oncology – A brief technical outline of the project. *Proceedings of the 2014 6th International Advanced Research Workshop on In Silico Oncology and Cancer Investigation – The CHIC Project Workshop (IARWISOCI)*. Athens, Greece 3-4 Nov. 2014. doi:10.1109/iarwisoci.2014.7034630.
6. Khoumbati K, Themistocleous M, Irani Z. Integration technology adoption in Healthcare Organisations: a case for enterprise application integration. *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*. 2005. Big Island, HI, USA. Available at: https://www.interaction-design.org/literature/conference/hicss-2005-38th-hawaii-international-conference-on-system-sciences. doi:10.1109/hicss.2005.331.
7. Krasner G, Pope S. A description of the model-view-controller user interface paradigm in the smalltalk-80 system. *J Object Oriented Program*. 1988;1(3):26–49.
8. Hardt D. The OAuth 2.0 Authorization Framework, IETF RFC 6749, October 2012.
9. Bizer C, Seaborne A. D2RQ – treating non-RDF databases as virtual RDF graphs. *Proc. 3rd International Semantic Web Conference (ISWC*04). Hiroshima, Japan. 2004. Available at: http://iswc2004.semanticweb.org/
10. The Web framework for perfectionists with deadlines | Django. *Djangoprojectcom*; 2016. Available at: https://www.djangoproject.com/. Accessed May 16, 2016.
11. Welcome to Tastypie! – Tastypie 0.13.3 documentation. *Django-tastypiereadthedocsio*; 2016. Available at: http://django-tastypie.readthedocs.io/en/latest/. Accessed May 16, 2016.
12. Homepage | Celery: Distributed Task Queue. *Celeryprojectorg*; 2016. Available at: http://www.celeryproject.org/. Accessed May 16, 2016.
13. RabbitMQ – Messaging that just works. *Rabbitmqcom*; 2016. Available at: https://www.rabbitmq.com/. Accessed May 16, 2016.
14. MongoDB for GIANT Ideas. *MongoDB*; 2016. Available at: https://www.mongodb.com/. Accessed May 16, 2016.
15. Models. Welcome to Django OAuth Toolkit documentation – Django OAuth Toolkit 0.10.0 documentation. Available at: https://django-oauth-toolkit.readthedocs.io/en/latest/. Accessed September 1, 2016.
16. Spanakis E, Kafetzopoulos D, Yang P, Marias K. MyHealthAvatar: personalized and empowerment health services through internet of things technologies. *Wireless Mobile Communication and Healthcare (Mobihealth), 2014 EAI 4th International Conference on Wireless Mobile Communication and Healthcare – Transforming Healthcare Through Innovations in Mobile and Wireless Technologies*. Athens, Greece, 2014:331–4. doi:10.4108/icst.mobihealth.2014.257500.
17. Georgiadi E, Dionysiou D, Graf N, Stamatakos G. Towards in silico oncology: adapting a four dimensional nephroblastoma treatment model to a clinical trial case based on multi-method sensitivity analysis. *Comput Biol Med*. 2012;42(11):1064–78. doi: 10.1016/j.compbiomed.2012.08.008.
18. Stamatakos G, Georgiadi E, Graf N, Kolokotroni E, Dionysiou D. Exploiting clinical trial data drastically narrows the window of possible solutions to the problem of clinical adaptation of a multiscale cancer model. *PLoS One*. 2011;6(3):e17594. doi: 10.1371/journal.pone.0017594.
19. Graf N. MyHealthAvatar Deliverable No. 2.2 Scenario Based User Needs and Requirements; 2013. Available at: http://www.myhealthavatar.eu/wp-content/uploads/2013/04/MHA_Deliverable_D2-2_Scenario-based-user-needs.pdf. Accessed September 1, 2016.
20. Dong F. MyHealthAvatar Deliverable No. 9.2 Development of Demonstrators; 2016. Available at: http://www.myhealthavatar.eu/wp-content/uploads/2016/05/MHA_Deliverable_9_2_Development_of_Demonstrators.pdf. Accessed September 1, 2016.
21. Graf N. MyHealthAvatar Deliverable No. 9.4 Demonstration of MyHealthAvatar; 2016. Available at: http://www.myhealthavatar.eu/wp-content/uploads/2016/05/MHA_Deliverable_9_4_Demonstration_of_MyHealthAvatar.pdf. Accessed September 1, 2016.
22. Forgó N, Kollek R, Arning M, Kruegel T, Petersen I. Ethical and legal requirements for transnational genetic research. *Eur J Int Law*. 2011;22(2):614–7.
23. Forgó N. MyHelathAvatar Deliverable No. D11.1 The ethical and legal framework of MyHealthAvatar; 2015. Available at: http://www.myhealthavatar.eu/wp-content/uploads/2015/03/MHA-Deliverable_D11.1_Legal_and_ethical_framework_final_version.pdf. Accessed September 1, 2016.
24. Django-sslserver 0.19: Python package index; 2016. Available at: https://pypi.python.org/pypi/django-sslserver/0.19. Accessed September 1, 2016.
25. MathWorks T. MATLAB Runtime – MATLAB compiler – MathWorks United Kingdom. Available at: http://www.mathworks.com/products/compiler/mcr/. Accessed September 1, 2016.
26. Argyri KD, Dionysiou DD, Misichroni FD, Stamatakos GS. Numerical simulation of vascular tumor growth under antiangiogenic treatment: addressing the paradigm of single-agent bevacizumab therapy with the use of experimental data. *Biol Direct*. 2016;11(1):12. doi: 10.1186/s13062-016-0114-9.
27. Argyri KD, Dionysiou DD, Stamatakos GS. Modeling the interplay between pathological angiogenesis and solid tumor growth: the anti-angiogenic treatment effect. *Proceedings of the* 2012 5*th International Advanced Research Workshop on In Silico Oncology and Cancer Investigation – The TUMOR Project Workshop (IARWISOCI)*. Athens, Greece 22-23 Oct. 2012.
28. Williams T, Kelley C. *Gnuplot* 4.5: *An Interactive Plotting Program*. 2011. Available at: http://gnuplot.info.
29. Graf N. MyHealthAvatar Deliverable No. 7.1 Description of scenarios and use cases for MyHealthAvatar; 2014. Available at: http://www.myhealthavatar.eu/wp-content/uploads/2014/03/MHA_D7_1_Use-Scenarios.pdf. Accessed September 1, 2016.
30. Graf N. MyHealthAvatar Deliverable No. 9.1 Definition of the demos; 2014. Available at: http://www.myhealthavatar.eu/wp-content/uploads/2014/03/MHA_Deliverable_9_1_Demonstrations_define.pdf. Accessed September 1, 2016.
31. Kistler M, Bonaretti S, Pfahrer M, Niklaus R, Büchler P. The virtual skeleton database: an open access repository for biomedical research and collaboration. *J Med Internet Res*. 2013;15(11):e245. doi: 10.2196/jmir.2930.
32. Guy GRichardson L. Visit duration for outpatient physician office visits among patients with cancer. *J Oncol Pract*. 2012;8(3S):2 s-8 s. doi: 10.1200/jop.2011.000493.
33. Detmar S, Muller M, Wever L, Schornagel J, Aaronson N. Patient-physician communication during outpatient palliative treatment visits. *JAMA*. 2001;285(10):1351. doi: 10.1001/jama.285.10.1351.
34. Stange KC, Zyzanski SJ, Jaén CR, et al. Illuminating the 'black box': a description of 4454 patient visits to 138 family physicians. *J Fam Pract*. 1998;46(5):377–89.
35. Dong F. MyHealthAvatar Deliverable No. 10.3 Workshop disseminations. Available at: http://www.myhealthavatar.eu/wp-content/uploads/2016/05/MHA_Deliverable_10_3_Workshop_disseminations.pdf. Accessed September 1, 2016.
36. Graf N. MyHealthAvatar Deliverable No. 9.3 Report on the clinical acceptability and evaluation of MyHealthAvatar and recommendation. Available at: http://www.myhealthavatar.eu/wp-content/uploads/2016/05/MHA_Deliverable_9_3_Report_on_the_clinical_acceptability_and_evaluation_of_MyHealthAvatar_and_reccomendation.pdf. Accessed September 1, 2016.
37. Brazma A, Parkinson H, Sarkans U, et al. ArrayExpress – a public repository for microarray gene expression data at the EBI. *Nucleic Acids Res*. 2003;31(1):68–71. doi: 10.1093/nar/gkg091.
38. Baudis M, Cleary M. Progenetix.net: an online repository for molecular cytogenetic aberration data. *Bioinformatics*. 2001;17(12):1228–9. doi: 10.1093/bioinformatics/17.12.1228.
39. Christodoulou N, Stamatakos G. A modular semantic infrastructure layout for the management of hypermodel-pertinent metadata in the context of In Silico oncology. *Proceedings of the 2014 6th International Advanced Research Workshop on In Silico Oncology and Cancer Investigation – The CHIC Project Workshop (IARWISOCI)*. Athens, Greece 3-4 Nov. 2014. doi:10.1109/iarwisoci.2014.7034640.
40. Blazewicz M, Georgiadi E, Pukacki J, Stamatakos G. Development of the p-medicine oncosimulator as a parallel treatment support system. *Proceedings of the 2014 6th International Advanced Research Workshop on In Silico Oncology and Cancer Investigation – The CHIC Project Workshop (IARWISOCI)*. Athens, Greece 3-4 Nov. 2014. doi:10.1109/iarwisoci.2014.7034641.