

AN INFORMATION ARCHITECTURE FOR PHYSIOLOGICAL MODELS, CLIENTS AND DATABASES

C.F. Dewey, Jr.¹, B. Fu², S. Zhang¹, N. Dao³, W. Chuang², and Z. Li⁴

¹Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA USA

²Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA USA

³Health Sciences and Technology, Massachusetts Institute of Technology, Cambridge, MA USA

⁴Department of Bioengineering, University of Washington, Seattle, WA USA

Abstract- The comprehensiveness and complexity of physiological models will increase dramatically during the next decade. Advances in computer power, biological pathway information, information on molecular mechanisms, and new genetic data are all driving factors for these changes. This paper proposes an information system to support complex physiological information models. The system architecture requires three essential parts: a networked *Client* that generates a request for information; a *Computation Server* that performs the calculation; and a *Model Database* that provides detailed parameters to the Computation Server to facilitate solution. The Model Database is organized as an object-oriented hierarchy following the design used to store medical images and their metadata in accordance with the international DICOM standard for medical images [1]. A fourth critical component is also defined: a *Case Database* that archives the input data that has been used by the Client to define the case to be computed, and archives the output from the solution. This system is applied to a complex contemporary model of the electrophysiology of cardiac myocyte cells [the WIJ model; see [2] and [3]]. The resulting architecture has been demonstrated over the internet [ICMIT.MIT.EDU/myocyte_model.html].

Keywords - physiome, physiological models, databases, information architecture, DICOM, CORBA, XML

I. INTRODUCTION

Many of the quantitative models of physiology and biology that are embedded in scientific practice today were created to describe simplified physiological observations. The Womersley theory of pulsatile flow in a tube [4] was derived to describe blood flow in the arteries with a significant unsteady flow component caused by the periodic beating of the heart. Starling's law [see [5]] arose in an attempt to describe flow through heart valves and flow constrictions. The Horst and Cumings model [6] was an elegant way of describing the continuous statistical bifurcation of airways from the trachea to the small pulmonary bronchi.

The physiological models of today are considerably more complex for two reasons. First, the questions being asked are much more demanding. One is interested, for example, in the detailed distribution of unsteady fluid shear stress throughout a complicated arterial bifurcation, not the spatially uniform shear stress in a straight tube. Second, the level of

interdependence of the component models has increased dramatically. As an example related to the work reported here, a model of myocardial muscle cell electrophysiology is required before a realistic model of the actual performance of the heart can be developed. Frequently the variety of models required to answer a single physiological question exceed the expertise of any individual group of researchers, and one is required to interface different models from many different sources, each with different local design rules.

This paper proposes a specific information architecture to serve complex physiological information models. It is designed to facilitate the use of models being run on remote computation servers that receive input parameters and return computed results. Internet protocols are used to connect the *Client* and the *Computation Server*. It is found that robust object-oriented access to databases is necessary to store the parameters for the computation as well as to memorialize and make accessible the computational results. These various requirements have many analogs in previous medical and biological informatics problems such as the storage of multimedia medical images using the DICOM international format [1] and the development of gene expression databases [see www.ncgr.org/research/genex/other_tools.html for a current list of such repositories].

The next Section describes the design requirements for a computational architecture to support remote model execution. Then the individual parts of the architecture are discussed in some detail, with specific choices given for the software components used to develop each part. Finally, the architecture is applied to the specific case of a model of the cardiac myocyte that has recently been published. This model is representative of those that will be required as we approach grand problems such as the Physiome Project [7].

II. COMPLETE ARCHITECTURE

A general architecture for supporting remote computation facilities should be designed to serve many different physiological models. It should include the elements and tasks common to many models and encourage the reuse of software from one model to another. The application-specific part of the effort should be minimized. Previous experience with two specific models [1,8] has been used to develop the current design that attempts to achieve such a generalized architecture.

The network-based architecture includes four primary elements as illustrated in Fig. 1: a Client; a Computation

Server; a Case Database; and a Model Database. The objectives to be met by each of these components are outlined below:

1. The **Client** should run across the network using standard internet protocols. Specifically, we want the Client to run within a machine-independent browser, using downloadable interpreted programs for display and client-side functionality. The Client is used to set up the details of a particular information object that is called a **case**. Each case will contain unique input information that describes the problem to be solved and unique output that should be stored for future reference.

2. A **Computation Server** is a system that handles many different models. The server that we use as an example of the next generation of such software has been derived from the one developed by the National Simulation Resource at the University of Washington [nsr.bioeng.washington.edu]. The server consists of three parts (see Fig. 1). The first part is called the **Request Interpreter**; it handles the communication tasks and parses the input from the Client. The second part is the **Computation Engine**. It compiles and executes the runtime code from a list of appropriate equations, constants (reaction rates, boundary conditions, initial conditions, etc.), and computational parameters (convergence, time steps, etc.) supplied by the Request Interpreter. The final part is the **Results Output** module that opens communication channels with the Client and the Case Database and delivers results formatted for storage and display.

3. The complete set of information required to define a computational program that will be run by the Computation Server is called a **model**. The **Model Database** contains much of this model information so that all of the many parameters and constraints do not have to be entered by the client at the time a particular case of the model is run. The specified equations are stored in a format that can be parsed and assembled by the Computation Server. Each model includes one complete set of appropriate constants, such as reaction rates and initial concentrations, which is called the **default set**. Individual values within the default sets that are stored in the Model Database can be overridden by values specified by the Client at runtime. In actual operation, most cases are generated by specifying the model (and its associated default set) and then changing one or more parameters by specifying them as **default overrides**. The Request Interpreter receives data from the Client and assembles the complete case to be executed from the default parameters stored in the Model Database and the default overrides that are furnished by the Client. In these transactions, the Request Interpreter becomes a Client to the Model Database.

4. The **Case Database** is used to record the inputs that define each specific case and the results of the corresponding computations. The Case Database is designed to support robust correlation of the solution data among the large number of cases that are run. This query capability will be particularly important to the next generation of physiological models.

III. INDIVIDUAL COMPONENTS

Implementation of the network architecture defined by Fig. 1 can be accomplished using a variety of different software tools and algorithms. The choices that are made for the individual software components will strongly influence the ability to reuse the solution for different models and different computation engine designs. In the following paragraphs, we give the details of the design that was implemented and some rationale for the specific choices that were made.

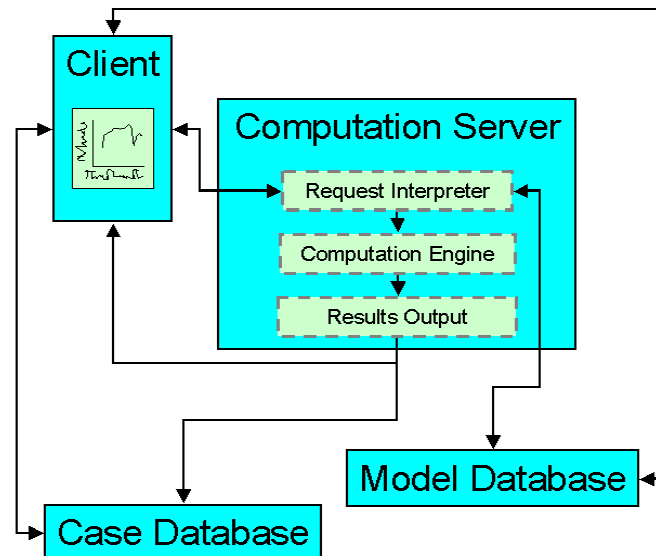


Fig. 1. The network architecture showing all of the key components of the complete model.

A. Client Design

The Client is a "thin client" built around industry-standard browser technology [Netscape Navigator and Microsoft Internet Explorer]. By "thin client" we mean that the Client programs that are executed on the browser are downloaded over the internet from the Server¹. The browsers support the Java language that is employed to build the user interface. The user interface receives and checks the syntax of the user commands and provides graphing capabilities for viewing the computational output. An example of the Java-based client window is given in Fig. 2.

The Common Object Request Broker (CORBA) standard [Object Management Group; www.OMG.org] is used to create access to the internet. The CORBA standard specifies a rich environment of object-oriented services that can be employed, but we have chosen to use it only as an efficient means to standardize connections across the internet. The two partners in each communication pair (the Client and the Computation Server, for example) both use CORBA

¹ It is possible to invoke mechanisms to cache frequently-executed programs in a Client and thereby reduce the overhead of retransmitting the programs at each invocation.

protocols, and therefore the same software can be used with all the different communicating entities including the databases. CORBA's communication access is through the Internet InterOrb Protocol (IIOP), a higher-level standard that is much more robust than HTTP. IIOP is also based on the ubiquitous low-level protocol TCP/IP. We have found that limiting the CORBA services to this simple subset avoids many of the painful experiences that have been reported in previous uses of the standard [9].

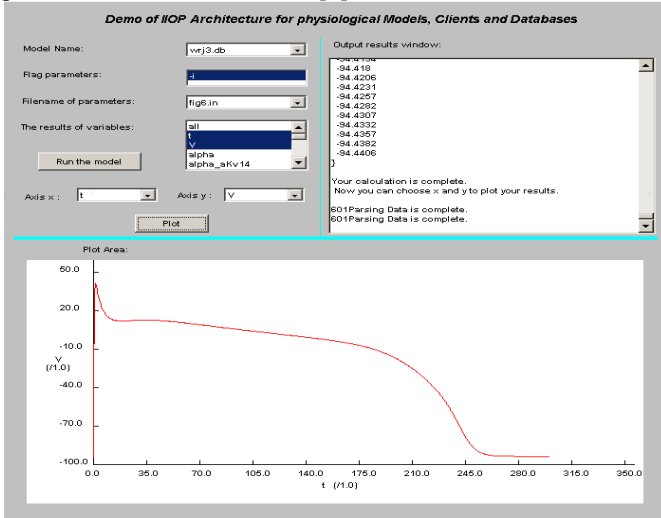


Fig. 2. The Client Java applet window showing all of the key components of the client functionality.

Information that is exchanged between the Client and the Computation Server is encapsulated as multimedia information objects using the Extensible Markup Language (XML) standard developed for the World Wide Web [www.w3.org]. XML is very useful here for several reasons. First, the information objects can be self-describing; they can contain not only cryptic numerical and text information and bitmaps but also instructions for interpreting these objects. This makes it very easy to build software to use the information contained within the XML objects. Second, the increased use of XML has produced many new tools that can be used to parse and manipulate XML-based messages. Finally, XML is based on interpreted ASCII text so that compiled software can be avoided and the parsing routines can run on many different computer platforms.

B. Computation Server

The Computation Server is assumed to service many models. This is the most general case and the one implemented here. Most current physiological modeling software runs on a single system and services only one model. In the future, a more general compute server architecture will be the rule rather than the exception.

The example that was used to represent the next generation of such servers is the computational engine developed by the National Simulation Resource at the University of Washington. This engine represents a set of

routines for solving coupled ordinary differential equations and compiling the run-time code from a list of the appropriate equations, constants (reaction rates, boundary conditions, initial conditions, etc.), and computational parameters (convergence, time steps, etc.). It has served more than 40 different models over the course of the last five years.

The specific model that was chosen for implementation was the action potential model for the canine cardiac myocyte published recently by Greenfield and colleagues [9] and is incorporated into the Winslow-Rice-Jaffri canine ventricular cell model [2]. The physiology defined by these two papers is called the WJR3 model. The default case consists of the published equations, model parameter values, and initial conditions. The modeling software and the WRJ3 model may be freely downloaded. See

<http://nsr.bioeng.washington.edu/Software/DEMO/CANINE-AP>.

Fig. 3 presents the dataflow architecture of the computation server. Originally, the computation engine was a monolithic entity that was coupled directly to an X-windows based user interface. One could run the application only by logging in to the server from a remote machine that supported X windows. All of the interaction at the remote site was transmitted directly to the server for parsing and interpretation and databases were not used. This was modified to support browser-based client programs that communicate with the server on a peer-to-peer basis using CORBA.

Fig. 3 illustrates that the same CORBA architecture is used to establish a communication link between the Computation Server and the Model Database. The Model Database can be located on the same physical machine as the Computation Server, or it can be located remotely and the requests and data transmitted over the internet. In all cases, the same CORBA interface architecture is used. This allows the different parts of the software to be developed and maintained separately. One Model Database could serve more than one Computation Server and vice-versa.

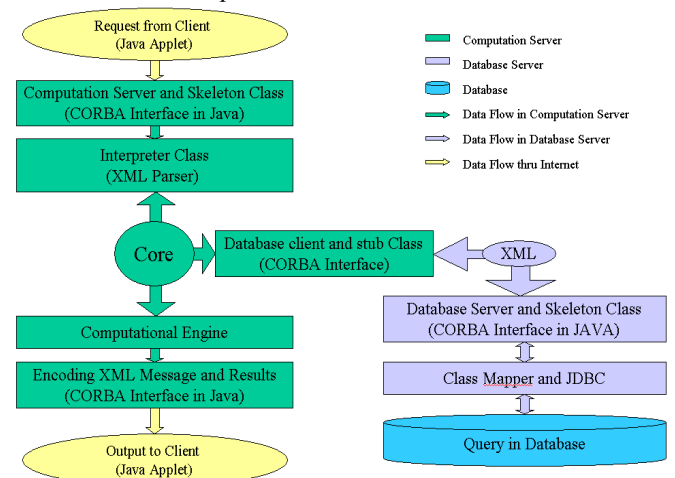


Fig. 3. Details of the internal architecture of the Computation Server. The request interpreter core handles the construction of the run-time parameters by finding the identity of the model, extracting the default parameters and

other information from the Model Database, and replacing the defaults with selected alternate (override) values sent by the Client.

C. Model Database

The Model Database is a key element in the system architecture. Its design influences the ease with which the entire system functions. It is the repository for all of the equations, the equation coefficients, the initial conditions, and other information necessary to fully describe the default models. Fig. 4 shows this database in schematic form.

Not shown in Fig. 4 is the Class Mapper [8,10] that holds the schema of the database and translates requests for information from an object-oriented form to the native language (e.g. SQL) of the database. The Class Mapper interface also serves as the interpreter and XML parser for requests from the Client. Because each database can potentially be from a different vendor and/or running under a different operating system, the ClassMapper must be customized to each situation. However, the view that the ClassMapper presents to the communication architecture (CORBA) is independent of the underlying database. Details are available in the references cited previously [8,10].

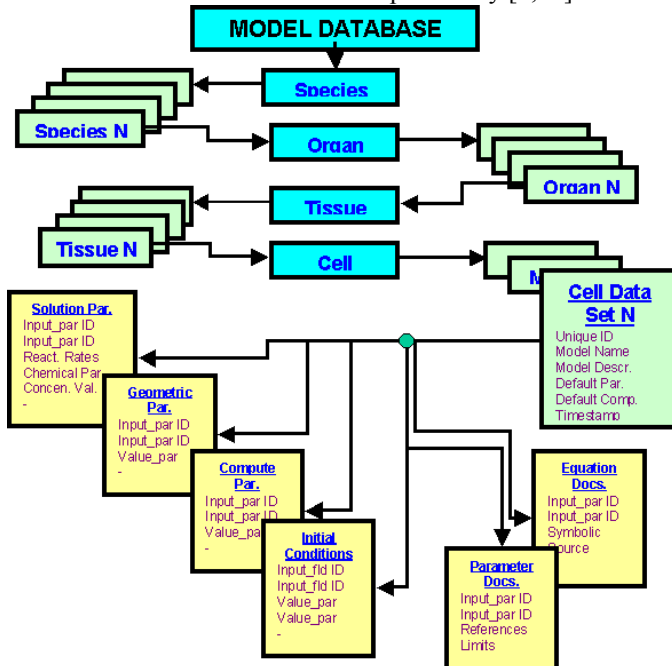


Fig. 4. The schema of the Model Database.

D. Case Database

The purpose of the Case Database is to provide archival storage for all of the parameters and output associated with the cases that have been processed by the Computation Server. In designs in which these data are not stored together, the results can become divorced from the parameters that were used to create them and the information becomes useless. It is also recognized that many models are computationally intensive; the solutions for a specific set of input values may take minutes or hours. Storing the output

will avoid having to run the case again. The database also facilitates correlation of the effects produced by varying one or more of the parameters of the problem.

The Case Database design shown in Fig. 5 was designed to accommodate a wide variety of models. To differentiate between different cases of the same model, the database is capable of storing either the complete default parameter set of the model as modified for the case that was run, or storing the model name and the default overrides that were substituted for the corresponding values in the default data set. In either case, all output from the Computation Server contains a unique ID of the case that was run along with appropriate tags to identify the name and version of the model that was executed to obtain the output. [For more consideration of the schemes for identifying and using multiple versions of physiological models, see [8].]

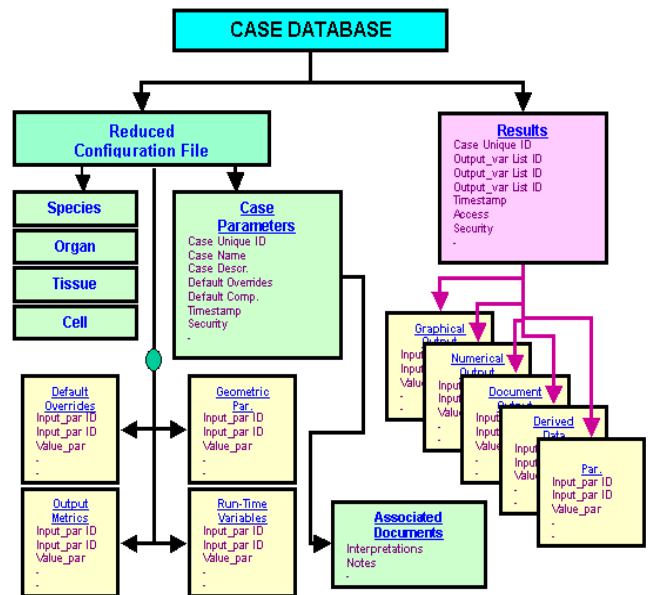


Fig. 5. The Case Database. This database is capable of holding all of the information about any case that has been run on any model supported by this architecture. It includes both input parameters and output results.

IV. DISCUSSION AND CONCLUSIONS

This paper describes a network-based computer architecture for supporting complex physiological models. It is based on the premise that these models will be sufficiently complex and difficult to maintain that they will be run only at selected sites on the internet. Therefore, access by many researchers will demand robust means for interacting with the Computation Server to initiate solutions and receive results. Current network browser software fits this requirement well. We have chosen to implement the communication path between Client and Computation Server using Java applets, CORBA communication utilities, and XML formatting for the data transmitted between the two. Our prototype can be run across the network from any internet browser by going to ICMIT.MIT.EDU/myocyte_model.html.

Once the primary information on a specific case is passed to the Computation Server, the server must assemble all of the data that are required for the computation. The first layer parses the commands and assembles the required information by acting as a client of a Model Database that is the repository of all of the information about the many models that are supported by the Computation Server. This design allows the data in the Model Database to be assembled, searched, corrected, and maintained independently. In some cases, the Model Database and Computation Server might reside on the same computer, but that is certainly not necessary.

There are many advantages to having a Case Database that will store the parameters of any case that was computed as well as the results of that case. First, the computations may be expensive, either because of per-case charges by the owners of the Computation Server or because of a very large computation time. Second, storing results and initial data in different places can frequently lead to disconnected information that renders both parts useless. It is much easier to develop tools that will carry case metadata along with the results and deposit both together in a single object-oriented multimedia database than suffer from maintaining synchronization between disparate databases. Third, the Case Database can be searched quickly to resolve discrepancies or to correlate output results. Our expectation is that the Case Database will be used extensively for cross-plotting data and performing discovery queries using standard database tools.

Anyone who has followed the evolution of software over the past ten years will recognize that the "middleware" components such as CORBA, XML, and Java are neither unique nor guaranteed to become permanent fixtures of the future. These components in our hands represent the best tools for developing distributed computing architectures at this time and in the near future. These components could be replaced by others without changing the basic design features that we find attractive. Java is used because it is relatively fast for the tasks that we expect of it, and it is easy to use. It is a complete object-oriented programming language that includes graphics tools and database capabilities. CORBA has managed to survive as an important middleware standard for half a dozen years, and there are both freeware implementations and commercial implementations of the standard. The network transport mechanisms are based on TCP/IP and can be built easily and reused. Finally, we encapsulate all the data sent over the network within XML wrappers. This supports the transmission of text, numerical results, graphical images, large data sets, and even supporting documentation in standard formats such as PDF. Using XML allows the packages to be unwrapped at the receiving end and parsed easily. It also allows the communication of methods and other information with the raw data.

We believe that architectures such as the one described here will be essential not only for physiological models but for molecular, genetic, and biological models as well. The medical and biological sciences are already awash in

information, and this state can only become more pronounced with time.

ACKNOWLEDGMENTS

We acknowledge the support of the National Simulation Resource Center at the University of Washington which is supported by the National Institutes of Health, the International Consortium for Medical Imaging Technology, and the Defense Advanced Research Projects Agency. Our thanks also go to Andrew McCulloch for several discussions, especially with regard to the design of the Client, to James Bassingthwaight for reviewing the manuscript and helpful insights into the use of this model with the Physiome project, to Patrick McCormick for help with design issues of the Class Mapper, to Raimond Winslow for assistance in the development of the original model of the WRJ3 cell computation, to Ken Fasman for his critique of an early draft of the architecture, and to Yuan Cheng and Jeannette Stephenson for technical assistance in writing the demonstration software and debugging the code.

REFERENCES

- [1].C. F. Dewey, Jr. and R. I. Kitney, "Creating DICOM-enabled clinical systems with robust querying capabilities," *Proc. Toward An Electronic Patient Record '98*, vol. 1, pp. 370-376, 1998.
- [2].R. Winslow, J. Rice, M. Jaffri, E. Marban, and B. O'Rourke, "Mechanisms of altered excitation-contraction coupling in canine tachycardia-induced heart failure. II Model studies," *Circulation Research*, vol. 84, pp. 571-586, 1999.
- [3].J. Greenstein, R. Wu, S. Po, G. Tomaselli, and R. Winslow, "Role of the calcium-independent transient outward current I_{to1} in shaping action potential morphology and duration," *Circulation Research* vol. 87, pp. 1026, 2000.
- [4].D. A. MacDonald, *Blood Flow in the Arteries*, 2nd Ed., William & Wilkins Company, Baltimore, 1974.
- [5].A. C. Burton, *Physiology and Biophysics of the Circulation*, 2nd Ed., Year Book Medical Publishers, Chicago, 1972.
- [6].Julius H. Comroe, *Physiology of Respiration*, 2nd Ed., Year Book Medical Publishers, Chicago, 1974.
- [7].J. B. Bassingthwaight, "Strategies for the physiome project," *Annals of Biomedical Engineering*, vol. 28, pp. 1043-1058, 2000.
- [8].N. Dao, P. J. McCormick, and C. F. Dewey Jr., "The human physiome as an information environment," *Annals of Biomedical Engineering*, vol. 28, pp. 1031-1042, 2000.
- [9].N. Greenfield, "CORBA's comeback," *Performance Computing*, pp. 11-17, April 1998.
- [10].P. J. McCormick, *Designing object-oriented interfaces for medical repositories. in: Electrical Engineering and Computer Science*, Cambridge, MA: Massachusetts institute of Technology, 1998.