Integrative Biology – exploiting e-Science to combat fatal diseases

D.J.Gavaghan[1], S.Lloyd[1], D.R.S.Boyd[1], P.W.Jeffreys[1], A.C.Simpson[1],

D.F.Mac Randal[2], L.Sastry[2], K.Kleese van Dam[2]

[1] University of Oxford, [2]CCLRC

Abstract

Heart disease and cancer are the two biggest diseases, and obviously the focus of intense work within the biomedical community. One aspect of this work is the computer simulation of whole organs based on molecular and cellular level models, and this requires the application of large scale computational and data management resources. The Integrative Biology project is a second generation e-Science project, funded by EPSRC, which will build on the output of first round projects and integrate these and other new developments into a customized Grid framework for running large scale, whole organ HPC simulations, managing a growing database of simulation results and supporting collaborative analysis and visualization. This paper describes the outline architecture and the set demonstrators being developed in the initial phase of the project.

Introduction

Approximately 60% of the UK population will die from either heart disease or cancer. Computer simulation of whole organs based on molecular and cellular level models offers the potential to understand better the causes of these conditions and eventually to develop new treatment regimes and drugs to reduce their threat to life. Realising this potential requires the coordinated use of computing resources on a scale not practically possible before the e-Science Programme; achieving this in practice is the main goal of the Integrative Biology project.

The project is a second generation e-Science project, funded by EPSRC, which will build on the output of first round projects and integrate these and other new developments into a customised Grid framework for running large scale, whole organ HPC simulations, managing a growing database of simulation results and supporting collaborative analysis and visualisation.

The three major cornerstones of the project are the cellular models of cardiac electrophysiology developed over many years by Denis Noble's group at Oxford, the extensive work underpinning computational modeling of the whole heart by Peter Hunter's group in Auckland , and Grid software already developed by several of the partners, particularly CCLRC. Figure 1 shows a view of a whole heart model. The long term goal driving the project is development of an underpinning theory of biology and biological function capable of sufficiently accurate computational analysis that it can produce clinically useful results.

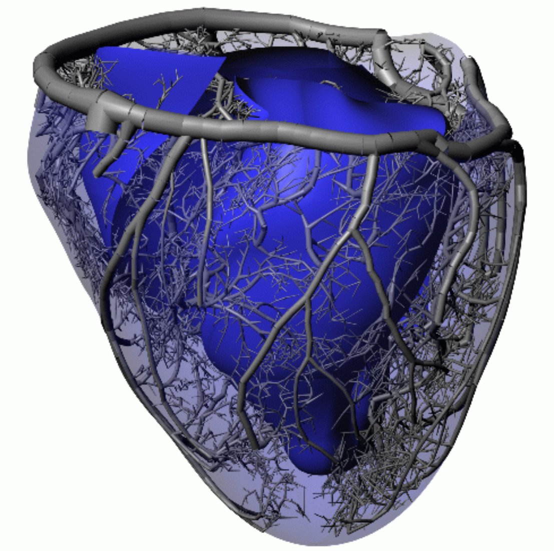


Figure 1: Whole heart model

Science challenges

The scientific agenda addressed by the Integrative Biology project includes:

- developing integrated whole organ models for some of the most complex biological systems in the clinical and life sciences;
- using these models to begin to study the development cycle of cardiac disease and cancer tumours;

Each of these groups have played a major role in setting up and promoting the IUPS Physiome project, see http://www.bioeng.auckland.ac.nz/physiome/physiome_project.php

- bringing together clinical and laboratory data from many sources to evaluate and improve the accuracy of the models;
- understanding the fundamental causes of these life-threatening conditions and how to reduce their likelihood of occurrence; and
- identifying opportunities for intervention at the molecular and cellular level using customised drugs and novel treatment regimes.

e-Science challenges

The e-Science challenges for Integrative Biology include:

- to provide transparent, co-scheduled access to appropriate combinations of distributed HPC and database resources needed to run coupled multi-scale whole organ simulations;
- to exploit these resources efficiently through application of computational steering, workflow, visualisation and other techniques developed in earlier e-Science projects;
- to enable globally distributed biomedical researchers to collaboratively control, analyse and visualise simulation results in order to progress the scientific agenda of the project;
- to maintain a secure environment for the resources used and information generated by the project without inhibiting scientific collaboration.

Beneficiaries

In both cancer and heart disease, Integrative Biology will assist the design and understanding of new drugs as well as enabling optimisation of novel treatments such as gene therapy or cancer vaccines which might complement conventional cytotoxic drugs. The tools developed by the project will improve the productivity of clinical and physiological researchers in academia and the pharmaceutical and biotechnology sectors. The UK e-Science community will benefit from access to new tools developed by the project and from the example of an integrated computational framework that the project will develop. This will be useful in other areas requiring a total system approach such as understanding environmental change processes. But most importantly, the ultimate beneficiaries will be patients with heart disease, cancer and, eventually, other potentially fatal diseases.

Project organisation and software architecture

The project brings together a team uniquely qualified to tackle the above challenges. It combines the biomedical and computing expertise of the universities of Oxford, Auckland, Sheffield, Leeds, Birmingham, Nottingham and UCL together with CCLRC and the support of IBM. Particularly close relationships exist with the RealityGrid, myGrid, Godiva, gViz, Geodise, BioSimGrid and e-DiaMoND projects; many of the investigators in Integrative Biology are also members of these other projects, and much of the technology developed within these projects will be of direct relevance to the Integrative Biology infrastructure.

The project team is organised into three main groups charged with developing:

- the modelling and simulation codes;
- the computational framework for simulation and interaction;
- the security infrastructure for the project.

Within these groups, cross-institutional teams are working on specific technical areas including heart modelling, cancer modelling, molecular and cellular modelling, testing tuning and running simulations, data management, computational steering, workflow, visualising data and user interfaces. Strong links have been established with other leading heart modelling groups across the world to widen the scope and impact of the science outputs, and channels to UK and international standardization bodies have been established.

Software Architecture

The ultimate end-users for the IB infrastructure are cardiac and cancer modellers, who not only want to run large complex models but also want to enhance and replace these models as their (and other’s) scientific understanding expands. At the same time, they have little or no interest in the underlying IT infrastructure, and certainly do not want to have to learn anything about it other than the minimum needed to use it. IB therefore has to provide a simple and coherent framework that hides the complexity underneath, while maintaining the flexibility inherent in that complexity.

The basic usage scenario is based on the notion that scientists want to carry out complex collaborative in-silico experiments much as they do currently with their existing codes and systems. IB should hide the complexity of the underlying IT infrastructure, but still make it easy to link to and exploit tools and techniques developed by others. Basically, the main tasks that the scientist wants to carry out are:

- *construct mathematical models of biological systems*. This may be a complete system or just part of a system, in which case the new model fragment has to be integrated into the existing model). These models have to be represented somehow, either as fragments of code or increasingly as definitions in the modelling language CellML.

- *build the model into a simulation code*. Normally existing off-the-shelf solvers that are suitable for the type of model being solved are used (but in some cases the scientist or a collaborating numerical specialist may wish to modify or replace the solver). In general solvers are optimised for certain classes of host, in particular to exploit parallelism on HPC machines.
- *specify a problem to be solved*. This mainly involves generating the input required to populate the model, defining the physical entity and any boundary conditions, setting up the initial state and specifying the solver's operational parameters and control mechanisms. Typically, this results in a set of files or references to data sources – see data management below.
- *initiate the simulation*. This can be quite complex, including selection of the computational resource(s) to be used, deployment of codes to these hosts, transfer of data to and between the hosts, co-scheduling and workflow management across possibly heterogeneous resources, etc. However, the scientist does not want to be exposed to any of this, other than making the trade-offs between accuracy, time, cost and risk where functionally equivalent alternatives are available.
- *control the simulation*. At the least, this includes the ability to monitor progress and abort jobs. It may also, depending on the sort of problem being addressed, include interactive computational steering of some simulation parameters. (Pre-defined computational steering for parameter space searching or for performance optimization are more problem specification issues rather than simulation control issues.) Ensuring the scientific validity of the steered simulation is left to the scientist.
- *manage and curate their data*. The obvious need is to be able to store and retrieve input and result sets without having to be concerned with the specific access mechanisms for each storage resource. For large datasets, the ability to pass references rather than copies is essential (as are 3rd party data transfers). Scientists also need to indicate what data to keep, and for how long. Naturally, data integrity and security are critical.
- *record appropriate metadata and provenance information*. To help in the location and interpretation of the data, a metadata catalogue or equivalent is required. While the system should automatically tag results with the relevant provenance data, the scientist has to be able to (or possibly be forced to) add supplemental semantic information.
- *analyse & visualize the results*. The main way of exploring results is through visualization, though other sorts of analysis are also important (especially comparisons with previous results and experimental data). Different simulations and different research questions will require different visualization – some of which are complex enough to warrant their own computational resource, set-up, steering, data management, etc (i.e. they effectively require all the steps described above but integrated with the main experiment).
- *collaborate with colleagues*. At any of the above steps (but especially for analysis and visualization) scientists may want to let colleagues see and interact with their experiment. (Security is again critical.)

Many of these components can be adopted from existing e-Science projects (as mentioned above), and if necessary adapted in collaboration with their original developers.

The above is a necessarily simplistic overview of what IB is trying to support. Not all steps will be required for every experiment – in some only data management and visualization are needed, in others only the input parameters vary from run to run.

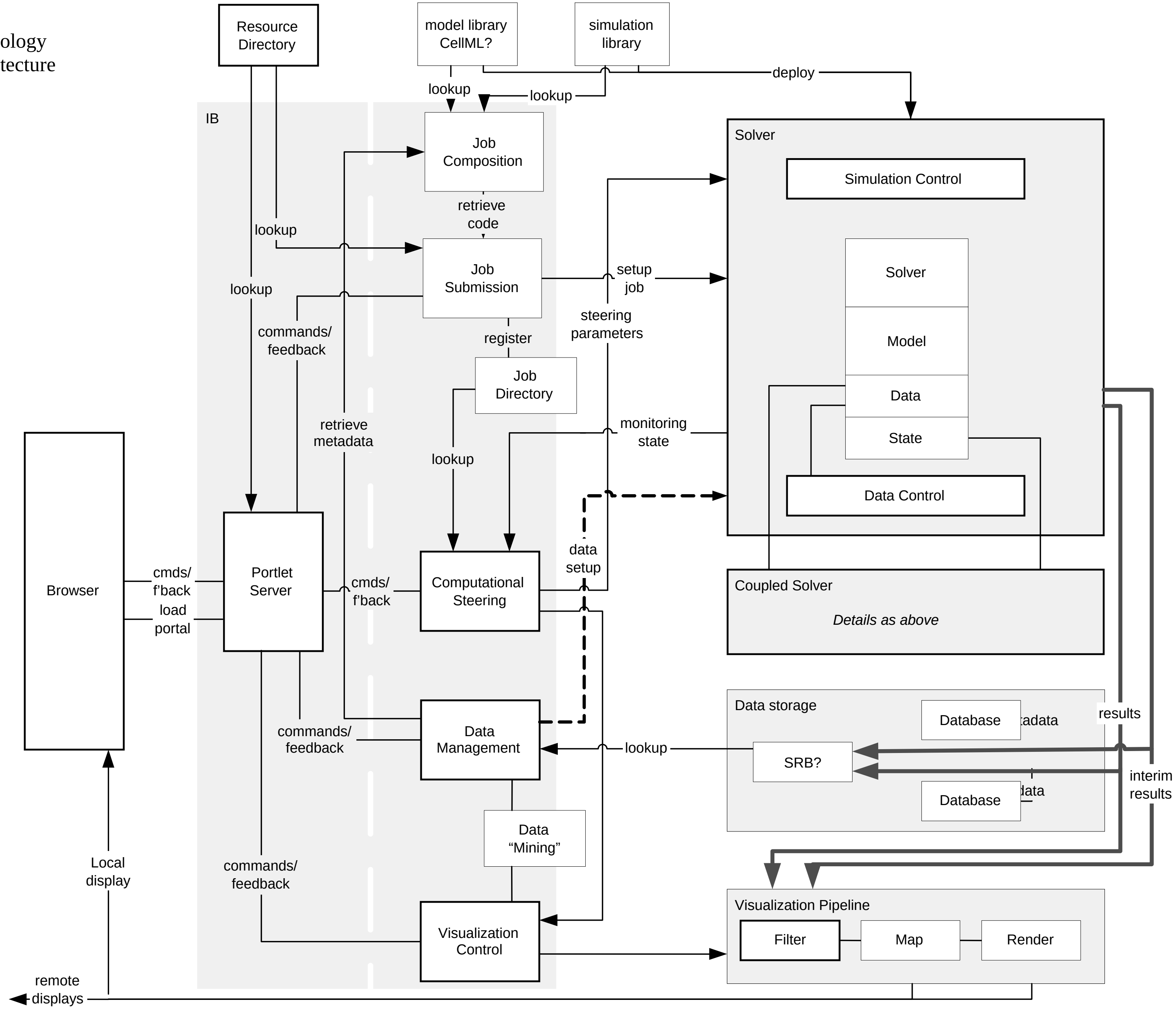
The development of the IB software architecture is heavily influenced by the experiences and knowledge gained within other e-Science (and science) projects and the requirements gathered from the IB end-users. (For more details of the IB requirements capture process, see the companion paper by Lloyd et al[1].) To this is added a close interaction with leading scientists, an experienced multidisciplinary team and a rapid prototyping methodology.

Underpinning the entire system are three overriding considerations:

- standardization;
- scalability; and
- security.

An initial outline of the architecture is shown in Figure 2.

Figure 2:
Integrative Biology
Outline Architecture



Outline structure

To support such a scenario and meet the four e-Science challenges mentioned earlier, the IB architecture breaks the system into five main parts, a portal, an IB infrastructure server and three specialist subsystems.

The use of portal technologies reflects a desire to minimize, if not eliminate, the mandatory footprint on the user's desktop. By only requiring a web browser, scientists using the IB infrastructure will be freed from having to maintain a defined local environment, and indeed freed to work from any location with the required connectivity. Whether this ideal can be achieved is still unclear – the two main areas that may require additional local support are security and visualization. Of course, users may choose to have some parts of the IB infrastructure hosted locally, for convenience or efficiency. In addition, portal technologies should make collaboration easier to set up and control. Current plans are to use the portal technology being developed in the Sakai project[2].

The IB infrastructure server provides, or provides access to, the required functionality described earlier. It has a dual function. Firstly, it wraps the IT complexity of the underlying systems and components, providing the scientists with a single straightforward way of carrying out their work, and secondly, it compartmentalizes the functionality in a "plug&play" fashion so that alternative or specialist components can be introduced without upsetting the "single straightforward way" the end users have adopted. Again, this is an ambitious target, and given the way scientists tend to want to squeeze the maximum performance out of any system they use, it is unlikely to be fully achieved; the best that can be expected is to ensure that the scientist only has to address the component being "squeezed", while the others can continue to operate and interoperate as normal.

The three specialist subsystems carry out the "grunt" work, number-crunching the simulation, storing and managing the data and visualizing the results. Though these will normally be "outsourced" to separate high performance systems, this is not mandated by the architecture. On the contrary, it is intended that the IB infrastructure could be used by a group using local resources to explore and develop models that can subsequently and trivially be targeted at the "big iron" available via the Grid.

The simulation engine will likely be one of a number of HPC resources available via IB. Functionality it will have to support includes discovery and status reporting, code deployment, job acceptance and management, computational steering, results management. Though the solvers required for different problems and resources will vary, the interfaces to the functionality identified above can be standardized, or at worst reduced to a small number of variants (and indeed some of this is already in place, c.f. resource discovery portals such as the CCLRC HPCportal[HPCportal]).

Results generated by simulations should be passed to a data repository where they can be properly managed and curated (the repository should also be able to handle experimental datasets). This repository will almost certainly not be a single system, rather it will be a virtual repository spanning many physical storage systems. It is vital that adequate metadata and provenance information is provided along with the datasets so that they can be easily and reliably retrieved for analysis or comparison. Systems such as the SRB[3], which is based round a metadata catalogue, provide the necessary facilities and are currently being integrated into the UK Grid.

Finally, a range of visualization systems will be required to allow the above datasets to be explored. Although different result sets (and different user requirements) need different visualization software and hardware, the basic underlying pattern of data filtering, mapping to a geometric form and rendering is common to most if not all visualization systems. By providing standard mechanisms for supplying data, controlling the filtering/mapping and directing the outputs, it should be possible not only to provide more visualization options for the investigator, but to support collaboration within geographically dispersed teams.

To keep the coupling between the components described above as loose as possible and to make it easy to select different components depending on the characteristics of the experiment being carried out, a Service Oriented Architecture is being adopted. Interactions between components will therefore use standard Web service mechanisms, augmented by Grid standards based on GT2 or WSRF where appropriate (and available). Thus most of the links shown in Figure 2 will be SOAP messages. Possible exceptions to this are the main data flow links between the solver, the databases and the visualization system. These links (the thicker arrows in Figure 2) will have to handle very substantial quantities of data, Gigabytes if not Terabytes, and more efficient Grid-based mechanisms would be required.

Security

The fact that security does not appear specifically in the Architecture diagram should not be interpreted as implying that security is considered to be something external to the IB architecture. Instead, security is both a property of the design of the whole system and a consequence of the operational constraints built into each component. The IB security model is being developed in parallel with the overall Architecture. The methodology followed is to first identify the assets, such as data, code, systems, configurations, etc. and second to examine the flows within the infrastructure, mainly of data and metadata but could also include code. From this the potential threats to be guarded against can be identified.

The IB infrastructure will sit in and coexist with a number of different environments, each of which will have its own security model. To ensure IB leverages on these, and does not duplicate (or conflict) with them, the boundaries of the IB security model have to be clearly specified.

In the proposed IB architecture, there are 4 such environments:

1. The user's point of entry.
   Whether this is a Web browser or a bespoke client, it will have the task of permitting the user to interact with the IB system and with "their" data and computations. As such, the integrity of the client and the security of information passed to or received from the client important. Given that the client (especially if a standard web browser) is owned by the user, and runs in an environment of the user's choosing, the client must be considered to be external to the IB security model. In other words, once data is passed to the client, its security is considered to be the responsibility of the user, and data received from the client is assumed to be arriving from a unprotected environment.
2. The Computational, Data Management and Visualization infrastructures.
   These are the machines on which the user's data is stored and on which their simulations are run. It can be assumed that these are servers run and managed by third parties, and hence providing and subject to their own security mechanisms. The exception to this is where the user specifies a specific machine to be used (typically their own) – in this case, a) that machine must provide the mechanisms necessary to interface with IB's security infrastructure and b) security on that machine is the responsibility of the user.
3. The IB platform.
   This is the machine(s) on which the IB infrastructure resides. It must provide the necessary infrastructure for the IB security model.
4. External sources.
   Users are almost certain to want to use arbitrary resources to provide data and programs. While the content coming from these can be considered to be "safe", i.e. deemed acceptable to the user, the source is considered to be external to the IB security model and therefore suspect.

The fundamental security unit in IB will be an "in-silico experiment", i.e. a set of users, resources, data and computations that form a coherent whole. (Perhaps experiments can be components of a study, which can be linked …; but that can come later). Each experiment has to be isolated or at least protected from interference from the outside and within the experiment the different components have to be allowed to interact without being able to compromise each other.

Access control for an IB experiment will be centrally managed by or on behalf of the PI for the particular experiment, who would specify who could do and see what. A sensible default would be needed, as well as a convenient interface to allow the PI to modify this default. Security enforcement will happen at the asset, or as close to it as possible, and will if possible be context sensitive (who is requesting what and why).

Finally, IB will provide mechanisms to permit users to specify which data flows, including third party data exchanges, are secured and to what level.

Demonstrators

In parallel with the development of the IB architecture, during the initial phase of the project a number of demonstrators are being developed. There are four main reasons for starting this so early in the project:

1. By providing an immediate boost in the size and/or complexity of problems that the scientists within the project can tackle, and thereby demonstrating direct benefits, the scientists buy-in to the project and its objectives will be strengthened.
2. By validating the initial architectural ideas, in particular the basic scenario outlined earlier and the gross decomposition of the infrastructure, the demonstrators will

highlight issues at an earlier stage in the architecture lifecycle.

3. By developing new systems, even in the absence of any middleware, the team gains an opportunity to learn and explore new technologies, and to evaluate what technologies / middleware would bring the greatest or fastest benefit to the scientists
4. By delivering tools and systems built with the Grid and the proposed IB architecture in mind, the scientists involved will gain a better understanding of the potential of these technologies and thus will be better placed to provide high quality requirements in the next iteration of requirements capture.

The planned demonstrators were chosen to span several qualitatively different biomedical systems.

The first demonstrator is targeted at the IB cancer modellers. This community typically has much simpler models that that of the heart, but are still interested in carrying out large parameter space searches. As the modellers typically use Matlab, it was decided to adopt the Matlab job submission features from the Geodise project, modify the job submission scripts to send a set of jobs to the JISC computational cluster at Oxford, target the data storage at the JISC data cluster at CCLRC and finally provide Matlab based visualization for the results. In this case Matlab would be providing the job composition, job submission and visualization functions of the IB architecture, while the actual simulations occurred on a HPC resource.

The second demonstrator is a soft tissue model used in image analysis for breast disease. This is written in C/C++. The code will be parallelized for MPI machines and computation steering facilities from RealityGrid will be added. The code will run on HPCx, with the visualization of results being carried out on the users machine.

The third demonstrator is the COR system, a 3D (anisotropic) finite difference non-linear reaction-diffusion model of the Sinoatrial Nerve. This is a single integrated code, handling model composition, simulation, data management and visualization, all written in Delphi for a Windows environment. The intention is to break this up along the lines of the IB architecture and provide a parallel solver thus enabling the users to exploit HPC resources to tackle significantly larger problems than they can on their existing hardware. One of the main features of the system is the ability to read and parse CellML files specifying different cellular models and from this build a complete model for simulation. Rather than rewriting everything, it was decided to leave the model composition in Delphi on its original platform, but submit the constructed model to a MPI solver running on a HPC resource. A standard visualization tool will be used in place of the bespoke code currently used, but the final images will be returned to the originating system in the same way as they are currently presented. The proposed architecture is shown in Figure 3.

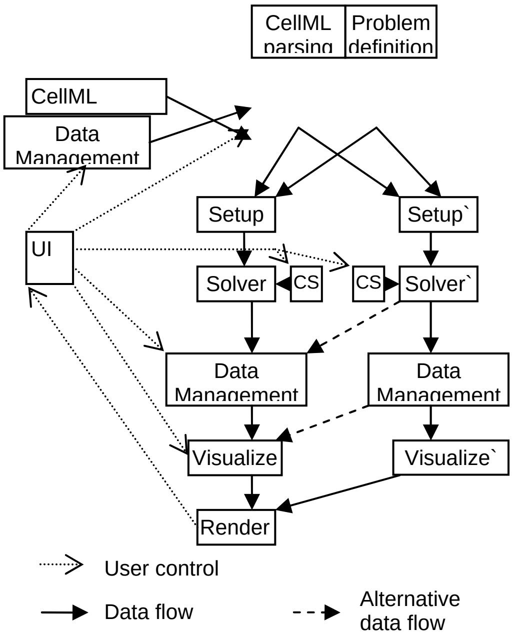


Figure 3: COR – Sinoatrial Node Model

The fourth demonstrator will take the CMISS modelling environment from Auckland. This provides job composition, simulation and visualization capabilities, organized as a GUI front end and a library of components to be composed into an executable. It currently can exploit multiprocessor machines but only on shared memory architectures. The intention for this demonstrator is to reengineer the solvers for MPI machines, introduce computational steering capabilities and provide links to a suite of visualization resources. This will result in production code, organized roughly along the lines of the IB architecture. The longer term intention is to extend this demonstrator into a full IB production facility.

References.

1. *S.Lloyd et al. Gathering Requirements for an Integrative Biology Project, All Hands Meeting, Nottingham, 31Aug-3Sept 2004*
2. *Sakai, Collaboration and Learning Environmetns, see http://www.sakaiproject.org/*
3. *SRB. Storage resource broker, see http://www.npaci.edu/DICE/SRB/*