Available online at www.sciencedirect.com
ScienceDirect
Computers and Chemical Engineering 31 (2007) 962–979
Computers & Chemical Engineering
www.elsevier.com/locate/compchemeng

# Addressing the challenges of multiscale model management in systems biology

J. Hetherington[a,b,d,*], I.D.L. Bogle[a,e], P. Saffrey[a,c], O. Margoninski[a,c], L. Li[a,b], M. Varela Rey[a,d], S. Yamaji[a,d], S. Baigent[a,b], J. Ashmore[f,g], K. Page[a,c], R.M. Seymour[a,b], A. Finkelstein[a,b], A. Warner[a,d]

[a] *Centre for Mathematics and Physics in the Life Sciences and Experimental Biology (CoMPLEX), University College London, Gower Street, London WC1E 6BT, United Kingdom*
[b] *Department of Mathematics, University College London, Gower Street, London WC1E 6BT, United Kingdom*
[c] *Department of Computer Science, University College London, Gower Street, London WC1E 6BT, United Kingdom*
[d] *Department of Anatomy and Developmental Biology, University College London, Gower Street, London WC1E 6BT, United Kingdom*
[e] *Department of Chemical Engineering, University College London, Gower Street, London WC1E 6BT, United Kingdom*
[f] *Department of Physiology, University College London, Gower Street, London WC1E 6BT, United Kingdom*
[g] *UCL Ear Institute, University College London, Gower Street, London WC1E 6BT, United Kingdom*

Received 1 December 2005; received in revised form 2 October 2006; accepted 6 October 2006
Available online 16 November 2006

**ostract**

Mathematical and computational modelling are emerging as important techniques for studying the behaviour of complex biological systems. e argue that two advances are necessary to properly leverage these techniques: firstly, the ability to integrate models developed and executed on parate tools, without the need for substantial translation and secondly, a comprehensive system for storing and man-ageing not only the models emselves but also the parameters and tools used to execute those models and the results they produce. A framework for modelling with these atures is described here. We have developed of a suite of XML-based services used for the storing and analysis of models, model parameters d results, and tools for model integration. We present these here, and evaluate their effectiveness using a worked example based on part of the patocyte glycogenolysis system.

*ywords:* Modelling; Systems biology; Metadata; Middleware

## Introduction

Modelling physiology is in many ways similar to the modling of process systems so there is much that chemical engiers can contribute. As with process systems, one of the major allenges in computational physiology is to efficiently inteate existing computational models which describe phenoma associated with a variety of spatial and temporal scales. ch models can be deterministic, stochastic, qualitative, or many other forms. An important part of this challenge is the storage, collation, and retrieval of models, along with their integration.

Our work (The UCL Beacon Project, 2002–2007) is part of the UK Department of Trade and Industry sponsored Beacon program, focused on harnessing genomics. We aim to build in-silico models that represent aspects of behaviour of the human liver, an epithelial organ. The methodology and modelling system should then be extendable to other epithelial organs. In building a fully integrated model of the liver, existing models of various components must be used along with newly devised models. Our approach is therefore to develop a system for the orchestration and integration of models. Not only will this system permit the development of integrated models which could not otherwise be constructed, it will also support the development of these models in a manner which increases the computational efficiency and relia-

* Corresponding author at: Room 203, CoMPLEX, Wolfson House, University llege London, 4 Stephenson Way, London NW1 2HE, United Kingdom. .: +44 20 7679 5076.
*E-mail addresses:* j.hetherington@ucl.ac.uk (J. Hetherington), ogle@ucl.ac.uk (I.D.L. Bogle).

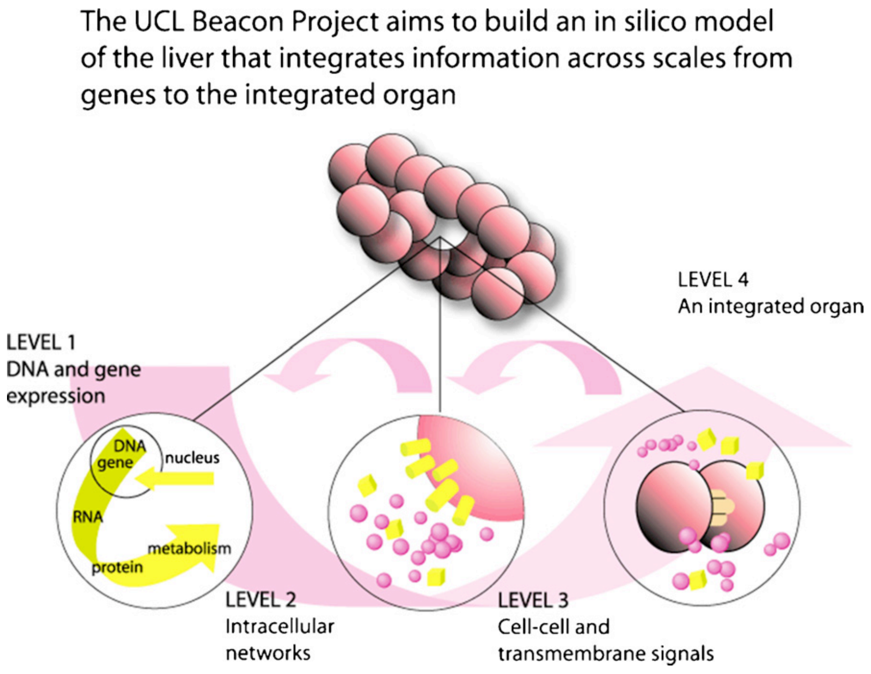

The UCL Beacon Project aims to build an in silico model of the liver that integrates information across scales from genes to the integrated organ
LEVEL 1
DNA and gene expression
DNA
gene
nucleus
RNA
protein
metabolism
LEVEL 2
Intracellular networks
LEVEL 3
Cell-cell and transmembrane signals
LEVEL 4
An integrated organ

Fig. 1. Crossing scales in biological modelling.

bility of those models, and reduces the time taken for such development.

The framework we have developed supports two key aspects of biological modelling: model integration across different scales, and the interconnection of the distinct components in biological systems. Interconnections are largely based on signalling i.e. the transport and reaction of chemicals between distinct components that drive the physiological system. Using this framework we aim in the project to develop a simulation environment in which a wide variety of models are integrated and exploited within a common domain of interest. These models may be at different levels of abstraction, may deploy different representations, and may focus on different interacting phenomena. Validation may give rise to model variants that will require management.

Our project will result in a system to integrate models addressing phenomena from the level of individual gene and cell features through tissue and organ models. Models at every level of the structure will be integrated, validated, and exploited using a plethora of mathematical, computational and experimental techniques. Fig. 1 shows the hierarchy of levels of signalling activity in many physiological systems.

One of the fundamental issues in model integration is how to handle the intrinsic inter-relationships between different models in an efficient way. Individual models are built up in an isolated biological environment relative to the real physiology and the purpose of linking different models is to recover the physiological conditions in terms of the context the models cover. Our computational framework for linking biological models will take account of the intrinsic couplings existing among the models, while allowing the flexibility that comes from being able to 'plug' in different choices of model, and link models which take different approaches to modelling, or which apply to different scales of consideration.

In this paper, we shall review existing work on computational infrastructure for systems biology, argue that two areas of software engineering (information management and encapsulation) should in particular be brought to bear upon the problem and describe a series of software modules we have authored that together constitute a complete computational environment for systems biology. In particular, the system supports the integration of models built in very different software environments while leaving the authoring and execution of the component models within those environments. We provide evidence for the effectiveness of our technique using an example model of part of the response of the liver to adrenaline, where one of the component models is built in Mathematica, and another in X-Phase-Plane-Auto (XPPAUT).

## 2. The state of the art

Much current modelling work in biology does not take into account the potential plethora of different models nor how to 'orchestrate' them. Integration mechanisms are at the program code level. A good example is the work on the heart carried out by Denis Noble and his team (Noble, 2002). Other groups are also attempting to take a more considered approach to model integration, and we review some related work here.

### 2.1. *Model management and process engineering*

Model management has been a topic of interest in process design for many years. Vazquez-Roman, King, and Banares-Alcantara (1996) developed a knowledge-based modelling support system which aimed to maintain the evolution of the model, support the development of understanding of the processes that are being modelled, and improve co-operation between modellers. More recently, Bayer and Marquardt (2004) discussed e many open issues of information modelling and presented conceptual framework for supporting the development and tegration of information models. The CAPE OPEN standard elaud, Pons, & Braunschweig, 2002) defines rules and interces that allow computer aided process engineering componnts to interoperate helping to facilitate the implementation standard interfaces between commercial tools used in the ocess industry. Engineering modelling approaches have been plied to hepatocyte cultures in artificial liver systems (Sharma, rapetritou, & Yarmush, 2005).

## 2. *Approaches originating in systems biology*

In recent years, several attempts have been made to create hemes, or frameworks, that would enable the exchange and tegration of models in systems biology. Broadly, these can be vided into three categories: modelling languages, modelling ameworks or tools, and tool integration frameworks, which ovide some support for model integration.

### 2.1. *Systems biology modelling languages*

Systems biology has seen the development of several XML-sed modelling languages designed to enable the representation d manipulation of biological models. The best known of ese are Lloyd, Halstead, Nielsen, and Bullivant (2004) and ucka, Finney, Sauro, Bolouri, and Doyle (2003). Both support ainly the representation of ODE-based models, and seem to designed mainly for modelling biochemical reactions at the llular level. In both languages, the mathematical details of the odel are specified using MathML.

SBML has seen considerable success in standardising the presentation and exchange of models, but is not focussed on tegration and does not allow the modular approach to modling we describe in Section 4. CellML also has a substantial dy of models represented, and in addition provides the means specify input and output variables for each model, allowing odels to be linked together. However, all the models linked ust be specified in CellML. Thus, neither CellML nor SBML ow for the integration of heterogeneous models developed in fferent languages and tools.

These languages are associated with attempts to handle the anagement and use of collections of models written in them: unter, Robbins, and Noble (2002) aims to collect together mods in CellML, categorise them and associate them with a small nount of static metadata. The Systems Biology Workbench oject (Hucka, Finney, Sauro, Bolouri, and Doyle, 2002), see low, is based around SBML.

### 2.2. *Modelling frameworks*

The E-Cell project (Tomita, Hashimoto, Takahashi, & imizu, 1999) is a whole cell and multicell simulation tool in hich the various biological entities being modelled are repsented as objects. It provides the means to quickly create odels in several schemes, such as diffusion–reaction, S-System d flux distribution analysis. However, it does not support e integration of models created in other tools. The scope of Cell is somewhat narrower than that proposed here, concentrating mainly on simulations of intra-cellular bio-chemical processes.

*2.2.2.1. The virtual cell.* Loew and Schaff (2001) is a graphical tool for creating and running Spatial PDE simulations of reaction–diffusion processes within cells. The Graphic User Interface enables the user to define the biological compartments involved in the modelled process, and then link them to actual cell images, in order to produce the required spatial data. The user is then able to define the chemical species and the reactions involved in the process, and their location in the various compartments. The tool automatically generates the relevant set of ODE's or PDE's for the model, and solves them, using a numerical solver. Support for importing and exporting Cell-ML and SBML models are planned.

*2.2.2.2. The XS-system.* Antoniotti, Policriti, Ugel, and Mishra, (2003) enables the construction of models from a set of building blocks corresponding to chemical reactions. This paradigm allows the speedy and intuitive creation of a variety of chemical networks, with the resulting model represented as a set of ODE's, specifying the rate equations for the various substances involved. A representation of SDE's (stochastic differential equations), timed automata or hybrid automata is also supported.

Where models are composed of sub-models, which may themselves be quite complex, the 'building blocks' used within the XS-System are fairly simple and limited in scope, and intentionally so: the developers of the XS system suggest to think of their system as the RISC (reduced instruction set computer) of systems biology.

### 2.2.3. *Tool integration frameworks*

BioSpice (Kumar & Feidler, 2003) is a collaborative project of American universities and research centres, financed by DARPA, whose aim is to create a tool-set for modelling dynamic cellular network functions. The collaboration aims to build a comprehensive software environment that integrates a suite of analytical, simulation and visualisation tools related to cellular systems biology. While the project aims to support many types of spatio-temporal models, multi-scale modelling and model analysis as well as simulation, it is unclear yet what systematic approach BioSpice would take in order to achieve the integration of heterogeneous models.

The Systems Biology Workbench (SBW) (Hucka et al., 2002) is a generic middleware for the integration of software tools, used quite often in systems biology. While it has facilitated the integration of many different tools and utilities, it was not designed specifically to facilitate the integration of models, and can be viewed as a generic middleware architecture, similar, for example, to CORBA (Vinoski, 1997).

## 2.3. *Simulation frameworks*

Several frameworks have been designed specifically to enable the simulation of heterogeneous components independent of scientific domain.

The high level architecture (HLA), (Kuhl, Weatherly, & Dahmann, 2000), is a general purpose architecture for simulation reuse and interoperability.

In HLA, a central service is responsible for advancing the simulation time in discrete time steps. Each component then advances its own simulation by this discrete time step, and updates the other components of the current values obtained.

The dynamic information architecture system (Campbell & Hummel, 1998) is aimed at supporting mainly discrete event simulations involving a large collection of heterogeneous entities, while also incorporating spatial data, such as the agents' location. In DIAS models carry the implementation of the object's behaviour, and communicate only with entity objects, which have a global scope, never directly with other models. We note that both in DIAS and in HLA only one algorithm (single time stepping) can be used to computationally integrate models together, while our suggested architecture enables researchers to combine models together using a variety of different algorithms and tools. Also, both DIAS and HLA do not facilitate any form of analysis of the models after simulation.

### *2.4. Data repositories*

As well as model orchestration approaches there is significant work in the cataloguing of biological information in databases.

Most of the information management effort in systems biology in recent years has concentrated on creating database repositories for the large volume of genome and proteome data accumulated in the last decade. There are several such repositories including those maintained by the European Bioinformatics Institute (EBI) and the National Center for Biotechnology Information (NCBI). Tools such as Zdobnov, Lopez, Apweiler, and Etzold (2002) and Schuler, Epstein, Ohkawa, and Kans (1996) offer consolidated access to these databases. These are invaluable for model validation and, as models approach the level of accuracy necessary for predictive purposes this usefulness will increase—and it will be necessary to link these to model management systems.

Extensive databases such as Kanehisa, Goto, Ogata, Sato, and Fujibuchi (2000) and Joshi-Tope, Gillespie, and Vastrik (2005) that contain information on the processes and entities that link the genome with cellular biology—gene and gene products, chemical compounds and reactions, pathways and networks. There is no information about reaction rates or about the experiments through which the information was obtained, except for paper citations in free text.

Comprehensive enzyme databases such as Schomburg, Chang, Hofmann, Ebeling, and Schomburg (2002) and The EMP Project (1999) list the reactions each enzyme is involved with, as well as numerical data such as reaction rates. They also provide the relevant paper citations. BRENDA (Schomburg et al., 2002) also contains organism specific information such as the source tissue and localisation.

Other databases include MEDLINE for papers www.ncbi.nlm.nih.gov/PubMed/, the National Biotechnology Service Information for gene sequences www.ncbi.nlm.nih.gov, the GenomeNet Database Service (www.genome.ad.jp) for genomic information and BioCyc (www.biocyc.org) for pathway/genome information.

None of these databases covers all of the data that was required to create and parametrise our model of hormone-stimulated hepatocyte glycogenolysis. More importantly, none of them was created with the aim of serving as a systematic database for the different parameter values to be used in modelling, and obtained either through experimentation or previous modelling. Even when numerical values are available, details about the precise experimental conditions or modelling assumptions under which these values were obtained is missing.

## 3. Metamodelling

In order to understand biological modelling, we have modelled the elements involved in model construction and validation, thus elucidating a biological metamodel. This comprehensive "metamodel" (Finkelstein et al., 2004), underpins the development of the tools presented in this paper so it is reviewed here.

The metamodel representation developed by the project shown in Fig. 2 uses an 'entity-relationship' (ER) modelling approach (first presented in Chen (1976)) and presents an entity class (of objects). The lines between boxes represent relationships which are associations between entity classes, see Fig. 2. Each entity class may have attributes. Each relationship has a cardinality that represents the number of entity instances that can be tied together by an instance of the relationship.

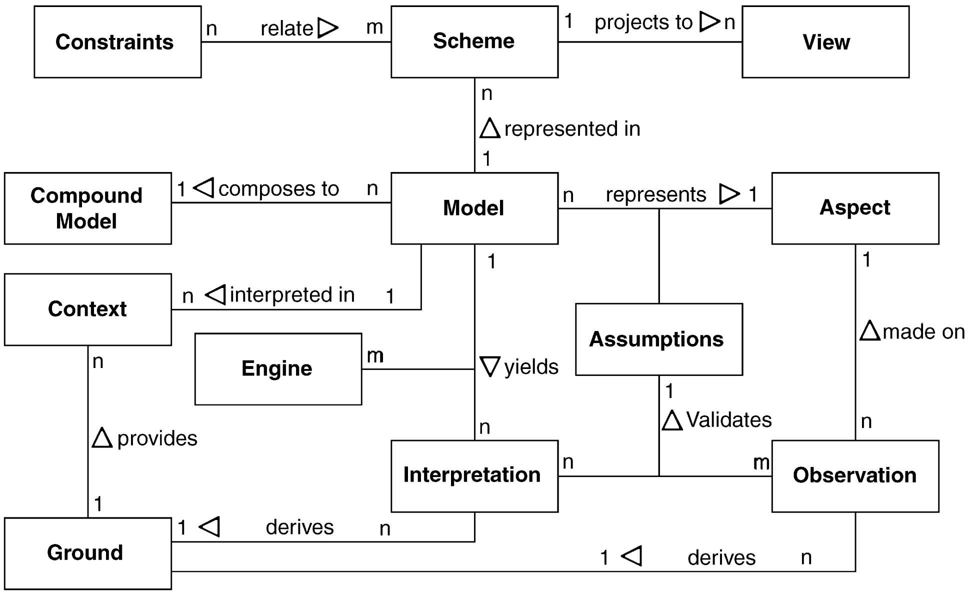
At the centre of the metamodel is the biological model itself. A model represents an aspect, the biological phenomena which is under study in the presence of a number of assumptions. The model is based in a particular scheme, the modelling paradigm that has been used, for example ordinary differential equations. Each scheme may have a number of views that show the content of the model, for example the presentation of the equations. Each scheme may also have constraints, that limit the model's ability to interact with other models.

A model is analysed or interpreted by an engine, such as a simulation tool, and in the presence of a context, extra data that parameterises the model. This analysis yields an interpretation, the results of a model. At the bottom of the diagram are the biological observations that provide the aspect for the model and validate model interpretations. Observations also provide the ground, the data upon which context information is based. This will be discussed in greater detail in Section 6.4. Finally, models can be composed to give rise to compound models, which will be discussed in greater detail in Section 4.

We have used this metamodel to organise our understanding of the biological modelling problem, in particular the integration of biological models which may be of many types of mathematical and computational formulation.

## 4. Modularity

We construct biological models by connecting together existing smaller models of individual phenomena. This approach has many advantages – if the component models are well understood and have been individually well-tested then much of this



g. 2. Metamodel entity relationship diagram. The ERD diagram convention is described in Chen (1976). The relationships depicted here are described in detail Finkelstein et al. (2004), and are summarised in the text. Note that our choice of terms deliberately abstracts many common modelling concerns, emphasising a neralised approach. Each box corresponds to an entity of concern in the modelling process, each line a relationship between them. The 1, . . ., $n$ notation is used to licate cardinality—so that $A_n \rightarrow_1 B$ indicates that several entities $A$ have a relationship with a single entity $B$.

nfidence should carry over to the larger model. It also has sadvantages – there may be subtle incompatibilities between odels which invalidate their integration. Our approach to build-g software to support model integration has been to try to verage one of the oldest software engineering paradigms: mod-arity. In this section we briefly review the well-established vantages of a modular approach.

One way to make a complex system more manageable is to eak it down into modules. Considering a system as a number separate modules has a number of advantages:

) Modularity aids understanding by presenting a system in distinct functional chunks.
) A different group of scientists can work on each module, distributing the effort and the expertise.
) Modifications can be made to one module without affecting the others. If desired, a module can be entirely replaced.
) Modules may be reused as part of other projects; a library of models-as-modules may be gradually accumulated.

Modularity may also be a means to address the significant allenge of modelling across scales. Modules could repre-nt the same system at different scales and be integrated to ovide the behaviour of the whole system. Our framework dresses these advantages directly by providing tools for con-ructing models as modules and allowing them to be used gether.

One rigorous use of modules is termed the “component-sed” approach. Such systems have three further features:

- *Interfaces*. Components communicate with each other only through well-defined function calls.
- *Encapsulation*. Components do not depend on the inner implementation of other components, and may influence other components behaviour only through the provided interfaces.
- *Language and tool independence*. Different components can be encoded in different languages, environments and tools.

Component technology is used extensively in software engineering, and has facilitated the composition, development, management and maintenance of large software systems. Experience with large software systems shows that the use of components may bring additional benefits: the ability to make use of third party proprietary code and the possibility of running modules in distributed environments, as in high performance parallel computing architectures.

## 5. The need for information management

Another important and well-established software engineering paradigm has regard to the careful management of the information pertaining to an endevour—the field of information management. At the moment, there is little standard practise in how data is recorded for use in biological modelling. Parameters are collected from the literature and recorded in an ad hoc fashion using notebooks or small-scale computing solutions. The tools used to execute models are installed and configured in many different variations, again often with little documentation. Results obtained from models are harvested for publications but not always made available in a standard form to others, or associated with detailed information about the tools and settings used to obtain those results, in conflict with the scientific doctrine of repeatability.

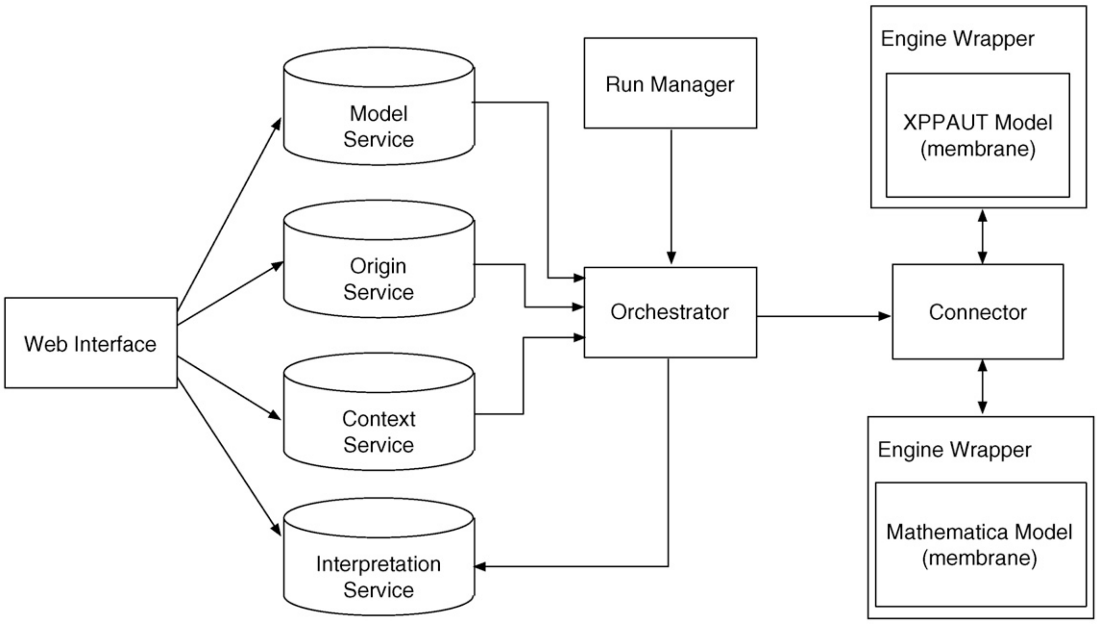


Fig. 3. Model integration framework. An overview of the principles of our integration framework—a connector to assist in creating consistent results for multiple models, wrappers to provide consistent access signatures for models, services to provide necessary information, an orchestrator to bring it all together, and an execution environment to expose these facilities to the user.

In general, there is a dearth of metadata, descriptive information about the data itself. What experimental results gave rise to a parameter value? What version of a simulation tool was used to generate a particular graph? What researchers chose one particular model structure over another and why? How has this model developed from its original conception (version control)? All of this information is useful in understanding existing work and taking it forward. The need for information management is discussed in greater depth in Finkelstein et al. (2004). A set of minimum standards for the metadata required to properly annotate a biological model has been defined (Novére, Finney, Hucka, & Bhalla, 2005).

Most biological modelling requires the selection of appropriate parameter values. Yet many authors do not emphasise the challenge of this area of modelling. One methodology for parameter determination is to find a least squares fit to data. Sometimes a proportion of values are taken from experiments. Projects with a lot of time and money will have enough on-site experimentation to parameterise their models—this is a rare but increasingly popular approach. In order for systems biology to be able to make good on its claim to be a child of the molecular biology revolution, it must be possible to obtain useful parameters by using data from existing published experiments. Modellers who follow this approach typically manage the information about where they have obtained such parameters either as comments to the model definition files or in their notebooks.

As well as allowing composite model execution, our framework aims to provide structure to manage the data used in systems biology. We encourage, but do not compel, a more disciplined and careful approach to parameter management. We shall make our management protocols clear in the remainder of the paper.

## 6. Integration framework

Fig. 3 shows an overview of our model integration framework, intended to facilitate a modular approach to systems biology modelling, with an emphasis on information management. Note that in Fig. 3, there are only two models. This is a simplified view, appropriate to the example model used later in this paper, see Appendix A.1. A composite model can possess much more complex topology consisting of many models and connectors—our framework has been used to support a seven-element composite model, discussed briefly in Appendix A.2.

At the core of the framework is an orchestrator, which reads the details of the composite model specification and mediates the communication between the sub-models, (by analysing the composite model specification and creating a computational representation of the input–output matrix, see Appendix B) and the deployment of these models on the appropriate model execution environments (by instantiating environment wrapper objects)—each environment is wrapped to translate the input and output for each model into an interchange format that can be manipulated by the orchestrator. The models are integrated by means of a connector which will be described in Section 6.2.

The framework is supported by a number of services that provide information about each model used during the integration and collect the results during a model run. This is a service-based architecture, familiar to computer scientists, and will allow us to harness existing work in web services (Christensen, Curbera, & Meredith, 2001). These services effectively provide databases for the information needed in the modelling process. A “context service” stores parameters, an “origin service” stores experiments and papers, an “interpretation service” stores results, a nodel service" stores models, and an "engine service" prodes execution environments to interpret the models.

We emphasise that our framework is modular not only in management of component models, but in other components the framework: connectors, services, and wrappers. Users of e framework are free to build their own components. In this ction we shall discuss briefly in turn the principles underlying ch of the concrete pieces of software which together constitute r modelling framework. More detail regarding each of these mponents can be found in three associated papers—in this per we hope to give an overview of how the components fit gether. Details of the numerical ("waveform relaxation") algohm used in the connector are given in Appendix B. Details of e XML formats used to describe model metadata and specify mposite model linkages are given in Margoninski, Saffrey, etherington, Finkelstein, and Warner (in press). Details of the formation management structure are given in Saffrey et al. 006).

## 1. Modelling environment wrappers and information terchange formats

Mathematicians and biologists use a variety of tools to rite their models including Mathematica, XPPAUT, MATLAB, epasi and SBML. Much effort has gone into translating model scriptions from one environment to another, or into establishg common standards for model description, see Section 2.2.1. e advocate an alternative approach: that of permitting existing odelling environments to talk to one another, through stanrd run-time interfaces. Thus, each modeller can work in the vironment they are used to.

In addition, we avoid the error-prone step of model translan. Of course, this functionality should be supported by generic iddleware systems. However, to enable this functionality, it necessary to specify standard "function signatures" for the ays that models can be accessed (such as the standard ratest-values-in interface for ODE models, which we call a "rate lculator" interface) We currently use C++ pure-virtual classes define these interfaces, but are planning in the future to make e of web services such as SOAP (Box, Ehnebuske, Kakivaya, Layman, 2000) and XML-RPC (Winer, 1998–2003). Defition of standard interfaces also requires definition of standard ta models for the information to be passed. We term these stanrdised interfaces and data structures "information interchange rmats", and in our example use one for functions of time, fined as uneven-time-series with linear interpolation available an access method (Polytrack).

We have authored Mathematica and XPPAUT wrappers. The athematica wrapper uses MATHLINK, the Mathematica API, d, from the point of view of the modeller, looks exactly like riting an ordinary Mathematica notebook for the model. The odeller simply replaces an NDSolve call with a homologous ll to NDFramework, a function we have written to export a athematica model for use with our framework. The XPPAUT plementation patches the XPPAUT open source, and the moder need only add an additional line to the model definition file ecifying the variable names to be exported and imported from the framework. These two modelling environments are very different—XPPAUT is a more traditional ODE solution environment, with models authored as model definition files with parameters and model definition intermingled, while Mathematica is a sophisticated hybrid analytical/numerical mathematical investigation environment, whose "notebook" use-metaphor creates a very free-form, asynchronous, step-by-step approach to model definition. We believe that successfully wrapping these two very different environments to permit easy interoperability with a common call signature, both at run-time and in terms of their interaction with the information management services, is a significant demonstration of the effectiveness of our approach. In particular, we achieve this without significant loss of expressiveness within the language of either environment.

## 6.2. Model integration connectors

A critical component of our framework is the idea of model integration connectors, which allow two separate models to be executed together. A connector is a means to solve several models as one, and embeds mathematical or computational techniques appropriate to the models in question. Part of the functionality of a connector may be to perform a transformation between different modelling schemes. For example, it may generate discrete events from continuous data received from an ordinary differential equation (ODE) model and pass these into a separate discrete event model. A connector may also serve to numerically integrate two models of the same scheme, where such an integration is not trivial. Connectors can be conceived which would connect sub-models that are stochastic or based on Bayesian network modelling. Connectors could also be constructed to link heterogeneous models, for example a link between a stochastic and an ODE model. One example of a model connector can be found in Tomita et al. (1999). If the models expose interfaces of the rates-of-change-given-currentvalues form, then any ODE solving numerical algorithm can act as a connector. The solution of models connected in this way is considered in Takahashi, Kaizu, Hu, and Tomita (2004).

However, for some models this interface is not available or its use may be inefficient. What if each model both expects as input and produces as output a set of 'timetracks', (a Polytrack), variable values as a function of time? It is this example we have chosen to explore here, as it provides a nontrivial challenge for wrappers, information interchange formats and connectors. One way to integrate such models with Polytrack interfaces is via waveform relaxation—the connector that we use for our example (Burrage, 1995). Waveform relaxation is a method devised in parallel computing for distributed execution. It is designed to deliver efficiency improvements to systems with disparate time-scales, but is also effective as a means to execute composite models, designed and executed on different tools and in distributed locations.

The algorithm, as applied to the simple case of two models used in our case study for this article, proceeds as follows. Execution begins with either model. The inputs that are not yet available from the other model are seeded with some suitable start values. The model is then solved to produce a set of outputs. These outputs are fed to the other model, which is solved passing back its output values. This procedure repeats until convergence. For a formal statement of the algorithm, see Appendix B. Since the algorithm does not require any input or intervention during the model execution, it can be applied to any simulation tool, as long as it provides a Polytrack at the end of the simulation. The standard function signatures defined by the framework and implemented via the wrappers enabled the development of a generic implementation of a waveform relaxation connector. Note that for such an algorithm to be necessary, the models must be cyclicly interdependent, as in our example.

Waveform relaxation has been shown to have good convergence properties and to provide great efficiency benefits for certain types of system. A full paper describing the application of this technique in systems biology is in preparation.

## *6.3. Model service: model metadata files and composite model specification*

For each model, XML files associated with the model definition specify the model's inputs and outputs—its required parameters and driving functions and its results. This includes a specification of default parameter values. We also provide tools to assist in the generation of these files for existing models. The schema is given in Margoninski et al. (in press), and is compliant with the standard laid out in Novére et al. (2005). Note that while the model definition itself is contained in a model definition file in the native language of the appropriate model authoring and execution environment, stored in a standard location, attendent XML files store associated information. This paradigm for the model database—a collection of models defined in native file format each associated with an XML file linked to an XML database, is repeated for all the database elements of the framework. This provides significant ease-of-use advantages with respect to those systems which attempt to mix metadata and model definition in a single file. (We believe that while XML is an appropriate language for the specification of rich, complex metadata, it is too cumbersome to be used as the basis for storing and authoring models.) Not all model dynamical variables or parameters need be exposed to the framework—some may be left defined only in the native model definition, permitting gradual adoption of the framework approach by nervous modellers. Model metadata files include information on the biological relevance and mathematical formalisation of the model interfaces, intended for human use, to facilitate the process of ensuring models to be linked are compatible. These can make use of formal ontologies to ensure consistent nomenclature. Model metadata files also include information on the computational implementation of these interfaces.

Information as to how models should be composed to produce a composite model is stored in a composite model specification. This file contains a list of models and how they are connected (the model topology), and also the metadata information attendent to the composite model—additional data required to execute the composite model, and what is produced (the inputs and outputs). Thus, the composite model is a model like any other in the system, and we anticipate that models may be recursively composed (we are currently engaged in modifying the orchestrator to support this.)

## *6.4. Context service*

We provide a database which stores the appropriate parameter data for models, and a variety of systems to make the database accessible to both humans and computers.

Each parameter is, in brief, stored with the following information:

- The ontological name of the parameter. An ontological name is a name based on standard ontologies, such as the gene ontology (Ashburner, Ball, Blake, & Botstein, 2000). This name should provide a unique identification for a parameter that can be recognised by other scientists.
- The name of the person who originally recorded the parameter.
- The category of the parameter, such as a rate constant.
- The possible values for this parameter. There may be multiple entries for a parameter value, each of which includes the value itself, the origin of the parameter (paper, experiment, estimate of a scientist or combination of these—see Section 6.5) and the confidence in this value.
- Further notes about the parameter.

In our implementation, we use XML to encode the parameters. This allowed us to make use of the wealth of tool support for XML, as well as its core features of extensibility and flexibility. We use a native XML database (Meier, 2002) to store and search the parameters. Detailed information regarding the formal specification as an XML schema is given in Saffrey et al. (2006).

In addition to the database itself, we have implemented graphical interfaces to this database that allow parameter metadata to be entered, searched and selected for use in models. The tools allow a complete set of parameters and their values needed for a composite model execution to be saved into a parameter run file. Functionality in the engine wrappers shown in Fig. 3 allows these values to be automatically inserted into each of the various sub-models at run time. Because the separation of parameters and model definitions is not always supported by the modelling environments being linked, several software layers must interact to achieve this. The orchestrator must determine which component model(s) the parameter applies to, reference the ontological name used for the parameter using the information in the component model CMSL files to determine the parameter's syntactic name for that model, and call the modelling environment wrapper's interface. The modelling environment wrapper must then adjust the parameter value—in the Mathematica case by scheduling an appropriate call to the Mathematica kernel to be evaluated after the model itself is loaded, and in the XPPAUT case by further dereferencing the parameter's syntactic name using the model definition file to find the numerical index for the parameter, and then modifying the appropriate array. Other tools understand the supported modelling languages and can ad unannotated model definition files to automatically allow e appropriate parameters to be added to the database and to e model metadata files (Section 6.3).

## 5. Origin service: experiment and other provenance anagement

The mathematical modelling of biology must support its nnection to the experiments which underpin it. One of the ost important aspects of this connection is the way in which perimental information is used to provide values for model rameters. It is surprising how often modelling papers fail to port the way in which their parameter values were obtained or w they are supported by experimental data—despite the fact at it is the difficulty of obtaining good parameter values which presents one of the biggest obstacles to the success of the odelling endeavour. Our tools provide XML databases, with propriate schema (again given in Saffrey et al. (2006)), to suport the archiving of lab experiments in a fashion which enables sy interrogation by both modellers and by the framework. The ML database scheme is consistent across the model, context, sult and provenance databases. Storing the data in an XML vironment allows us to query, link and present the data using e wealth of tool and language support for XML including rich ery languages like X-query (Meier, 2002). Web-interfaces for ch of the database services have been constructed. This system plicates some of the functions of laboratory information manement systems (such as ConturELN, Water eLab Notebook or OE2000) but is integrated with the rest of our framework and lored towards interdisciplinary results sharing. It is hoped in ture to test the use of our framework within a laboratory which s already adopted a LIMS system.

## 6. Interpretation service

An interpretation service stores the results of a model run. r an ODE model, results can be presented simply as plots variable values with respect to time but may also include ecific types of plot or further commentary. However, simple orage and presentation of time series results is inadequate—an terpretation service must support search over many collected sults, and complex analysis of these results. Each type of analis is an interpretation of a model and should be appropriately ored. Every interpretation should be linked back to its orins: the model itself and the parameters and configuration under hich it was run. This set of files should allow a model interpretion to be regenerated, for example, at a different location by fferent researchers. Our implementation of this service allows is, in a process we describe as "end-to-end" information manement. The framework stores all numerical data produced by odel runs in a consistent location, together with XML inforation explaining how this information is laid out, so that it can searched and queried. This design choice – the use of native tput data formats associated with XML metadata – means that e raw datafiles can be analysed by any bespoke model analis tools individual modellers may have become used to for eir component models, provides space advantages, and ensures that all model result data, even for models which have incomplete metadata descriptors, is archived in some form. This is an essential element preventing unnecessary model re-execution, while allowing gradual and easy adoption of the model metadata system. This is a form of "grey boxing", providing the correct balance of the advantages of full encapsulation ("black boxing") with the necessity to store all information in some manner.

The interpretation service also compiles automatic "model reports" bringing together in a clear fashion all model results, parameter values and origins.

## 6.7. Orchestrator

At the centre of the framework is an orchestrator, which loads the models, wrappers, and connectors, obtains information from the services, and manages the sharing of data between wrapped models. Key roles for the orchestrator includes interrogating the various model metadata and definition files to obtain the connection matrix for the WR algorithm (Appendix B), launching the various execution environments, and exchanging data with the run manager.

## 6.8. Execution interface

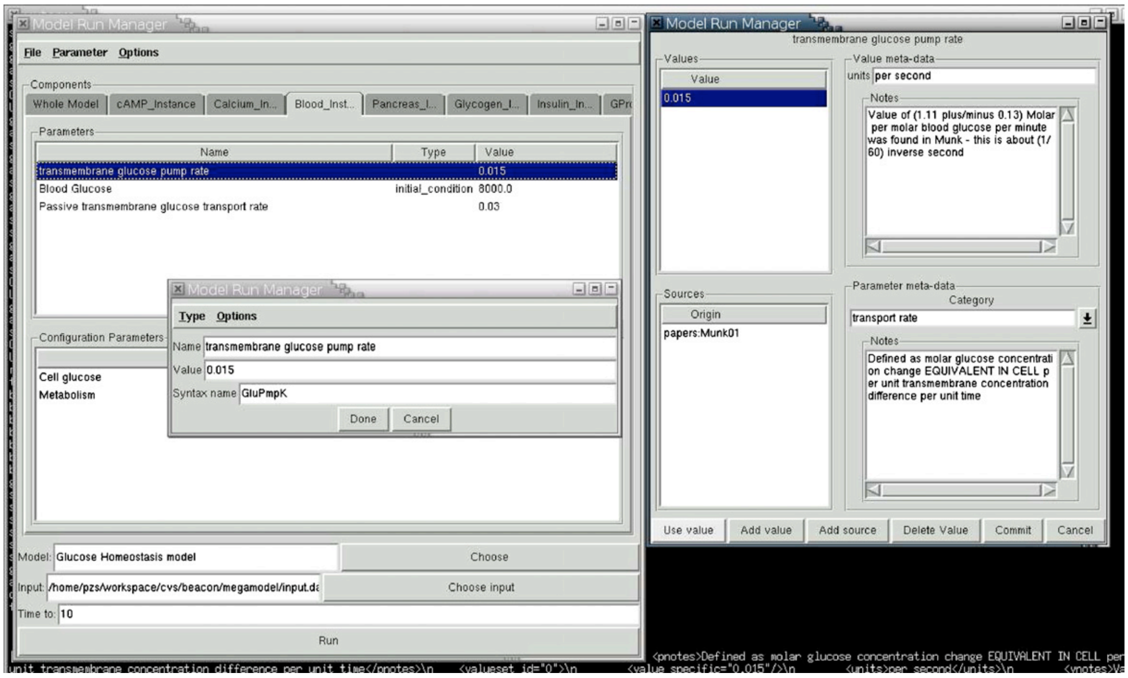
In addition to the web tools for interfacing with the database services, an installable application allowing more complex use patterns including the upload of new models to the system has been created. Model authors typically use this interface to launch runs of models on their local computer, and for this reason the programme is called the model run manager (MRM).

Fig. 4 shows a screenshot of this tool in the context of parameters being selected for a model run.

## 6.9. Summary of software components in the framework

We have implemented the following software elements to support this framework:

- Uneven-step interpolated timeseries interchange format—Polytrack.
- Uniform model call structure base classes for wrappers and connectors.
- Mathematica wrapper complying with our standard.
- XPPAUT wrapper complying with our standard.
- Implementation of waveform relaxation connector.
- XML-based parameter (context) database.
- XML-based experiment and paper (provenance) database.
- XML-based model result (interpretation) database.
- Web-based interface for managing our databases.
- Graphical interface allowing scheduling of model runs and selection of parameters from database—MRM.
- Automatic report generation based on result database.
- Miriam-compliant XML model metadata description file—model metadata format.
- Assistance for authoring model metadata files by analysis of model definition files in supported formats (Mathematica and XPPAUT).



Parameters

| Name | Type | Value |
|---|---|---|
| transmembrane glucose pump rate | | 0.015 |
| Blood Glucose | initial_condition | 8000.0 |
| Passive transmembrane glucose transport rate | | 0.03 |

Cell glucose
Metabolism

Name transmembrane glucose pump rate
Value 0.015
Syntax name GluPmpK

transmembrane glucose pump rate
0.015
papers:Munk01
units per second
Notes
Value of (1.11 plus/minus 0.13) Molar per molar blood glucose per minute was found in Munk - this is about (1/60) inverse second
transport rate
Notes
Defined as molar glucose concentration change EQUIVALENT IN CELL per unit transmembrane concentration difference per unit time

Model: Glucose Homeostasis model
Input: /home/pzs/workspace/cvs/beacon/megamodel/input.da
Time to: 10

```
<pnotes>Defined as molar glucose concentration change EQUIVALENT IN CELL per
unit transmembrane concentration difference per unit time</pnotes>\n    <valueset id="0">\n    <value specific="0.015"/>\n    <units>per second</units>\n    <vnotes>Va
```

Fig. 4. Example of activity with the model run manager (MRM). Along the top of the window are tabs indicating each submodel within the composite model. (In this case the seven-component example in Appendix A.2.) Below this are listed the model's parameters and "configuration parameters"—in this case some initial conditions. Highlighted is one of these parameters, the activity rate of a transmembrane glucose pump. Using the terms in the name (and in future we anticipate more sophisticated searches) an entry for the parameter in the context database has been found, with a candidate value, some associated notes, and an origin for that value which may be clicked on to find the appropriate paper or experiment. Options to set a global input function and to launch the model run are also indicated.

- XML description format for composite model specification—CMSL.
- Runtime management software capable of martialing models, engines, and parameters, and uploading results—orchestrator.

### 6.10. Example of use

We shall now further explain how the modelling process is carried out using our framework. This scenario is similar to that which occured during the development of our example model (Appendix A.1). In our two-model scenario one model ($A$) has been built by a user ($U_A$) aware of the framework, and another model ($B$) is to be connected to the first, but has been authored by a modeller independent of the framework ($U_B$). An experimental colleague of $U_A$, $E_A$ has provided relevant data for model $A$, while the parameters for model $B$ are obtained from a series of published papers by $E_B$ (Fig. 5).

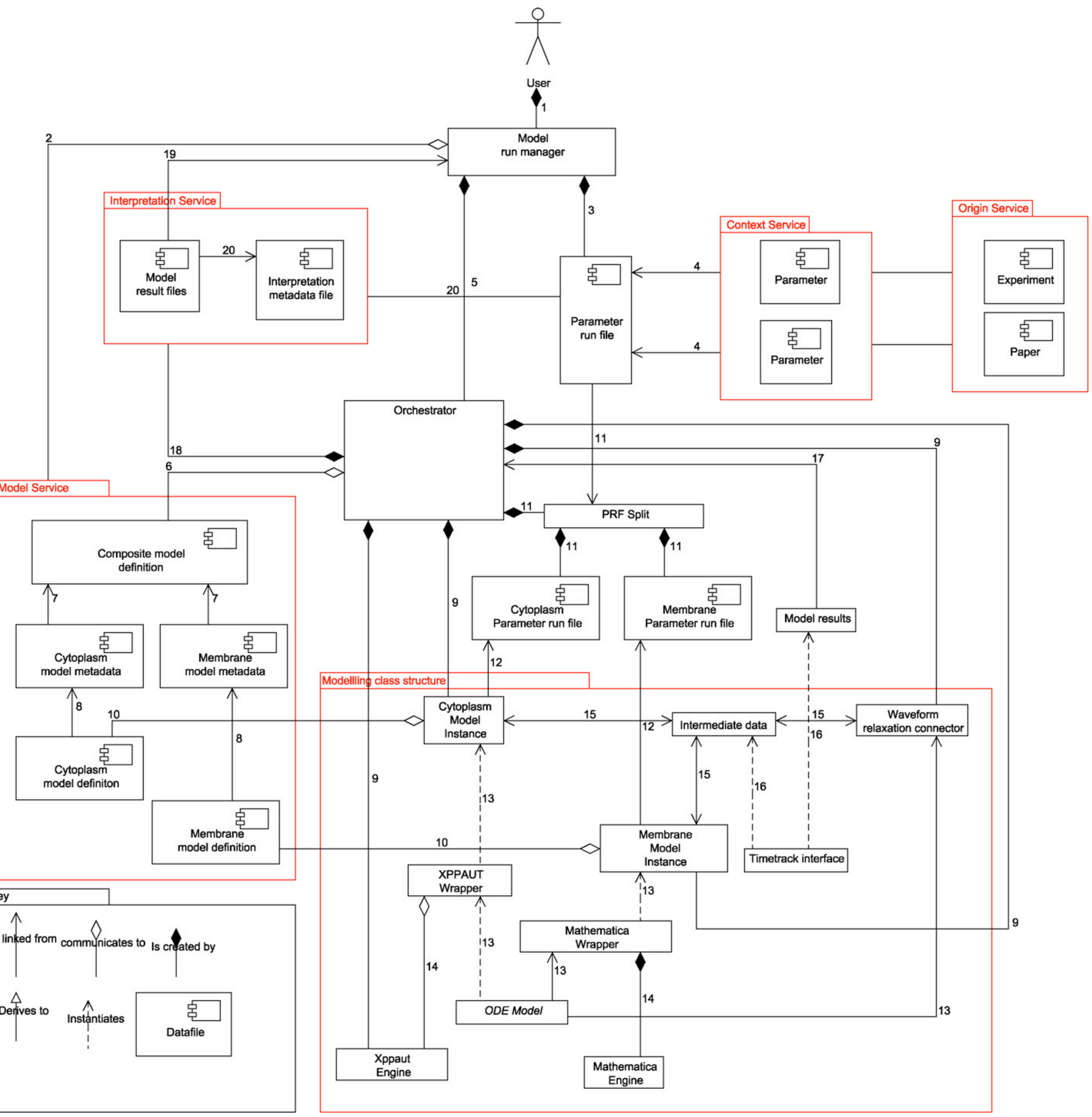
During the design process for model $A$, $E_A$ has uploaded a number of experiments into the origin service using the web interface. $U_A$ has created entries for each of the parameters of $A$ in the context service and appropriately linked these to $E_A$'s experiments.

In order to carry out the task of linking the models, modeller $U_C$ must make certain modifications to both models to ensure that the mathematical interfaces are compatible. To assist in this task, he authors model metadata files for the two models, defining the biological and mathematical interfaces for the models. Tool support assists in the addition of parameter metadata to these files, interrogating the model definition files to look for likely parameters, and the system will identify parameters in common between the models, to ensure that these have consistent values. $U_C$ also specifies the computational engines the models will run on, and makes adjustments necessary to the model definition files to ensure that these comply with the restrictions necessary to be compatible with the framework.

As model $B$'s parameters do not currently have entries in the context service, $U_C$ selects the parameters he deems most important and uploads these (using a component of the MRM which examines the model metadata file) he also adds the relevant publications by $E_B$ to the origin service. Additional parameters may be added to the service as necessary.

$U_C$ now creates a composite model definition file specifying how the models should be connected together and their order of execution. He launches the MRM and points it at this file, selects appropriate parameter values for his first numerical experiments, and begins the run. For the run-time flow of information, see Section 7.

Examining the automatically produced model reports from his run, he identifies parameters which require alteration and creates appropriate entries in the context service, identifying appropriate origins. After many runs, he may use XQuery to search the interpretation service for results which have been forgotten, determining the parameters resulting in each interpretation. Tool support allows results to be examined in a number of modelling environments, so that $U_A$ and $U_B$ can each view the model results in their own, different, modelling environments, using the user interface familiar to them.



g. 5. The modelling framework components, and their interactions. Note that, as illustrated in the key, this diagram is based on the notations of UML, but is not a mal UML diagram.

## Run-time information flow

The user launches the model run manager (1) and points it a composite model definition file (2). The user also chooses rameters, and the MRM builds from them a parameter run e (3) pointing to values in the parameter database (4). The RM launches an orchestrator (5), which uses the CMDL file ), to find (7) metadata files for the individual models, and, om them (8) the model definition files. It then instantiates (9) odels and their engines, based on (10) those definition files, and a connector. The orchestrator also breaks down the PRF (11) into individual model PRFs and uses these (12) to set the parameters for the individual models.

The individual models are based (13) on wrapper classes, which are in turn based (13) on a base model interface, as is the connector. Each wrapper is associated with a run-time engine (14). The connector and models exchange data (15) until the results are consistent. The results are based on a Polytrack format (16). The final results, in Polytrack format (17) are reported back to the orchestrator, which records them as a result file format

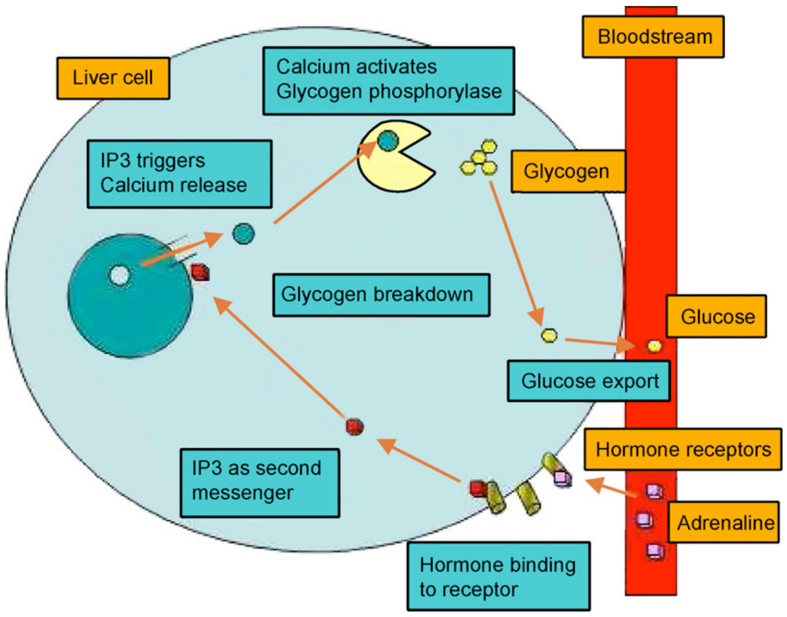


Fig. 6. The pathway by which adrenaline causes glycogen breakdown in hepatocytes.

(18). The orchestrator reports this file back to the MRM (19) which builds an interpretation metadata file referencing (20) the result file and the PRF file which made it.

## 8. Analysis of example

### 8.1. An example system

The system we have chosen to use to illustrate and test our techniques is based on existing models of hormone-stimulated hepatocyte glycogenolysis. This important physiological process is the means by which energy, in the form of glucose, is released from storage in the liver in humans and other animals. It constitutes one part of the glucose homeostasis system by which blood sugar levels are maintained within acceptable limits. Fig. 6 shows a cartoon of the main features of the pathway that controls this process. While we use a two-model exemplar here, we have developed a much larger (seven component) model of this system, which we will publish in due course.

Two pathways communicate the need for glycogen breakdown from the hormone receptors to the glycogenolytic enzyme, glycogen phosphorylase. The first is more associated with the response to glucagon, and acts through the production of cyclic adenosine monophosphate (cAMP), which activates an enzyme protein kinase A (PKA), which in turn triggers an enzyme-activation cascade leading to the activation of glycogen phosphorylase. The second is more associated with short-term response to adrenaline, and is the one modelled by our example in this paper.

In this process, activation of the hormone receptor results in the activation of a linked G-protein, which in turn results in the activation of phospholipase C (PLC) which results in the production of inositol trisphosphate (IP3). The IP3 moves from the cell membrane, where the processes up to this point have taken place, to the main part of the cell (the cytoplasm) where it then triggers oscillatory movement of calcium ions between the cytoplasm and the endoplasmic reticulum (ER), a cellular sub-compartment which stores a high concentration of calcium ions. During this oscillation, the higher average concentration of calcium in the cytoplasm results in the triggering of the sequence of enzyme activations leading to glycogenolysis.

Our case study uses the simple modular decomposition shown in Fig. 7. The two component models were developed in, and for the purposes of our experiment remain within, two very different modelling environments—Mathematica and XPPAUT. The first, ‘membrane’ module, representing the activation of a G-protein coupled receptor by a hormone stimulus, was built in Mathematica. Note that the G-protein coupled receptor is

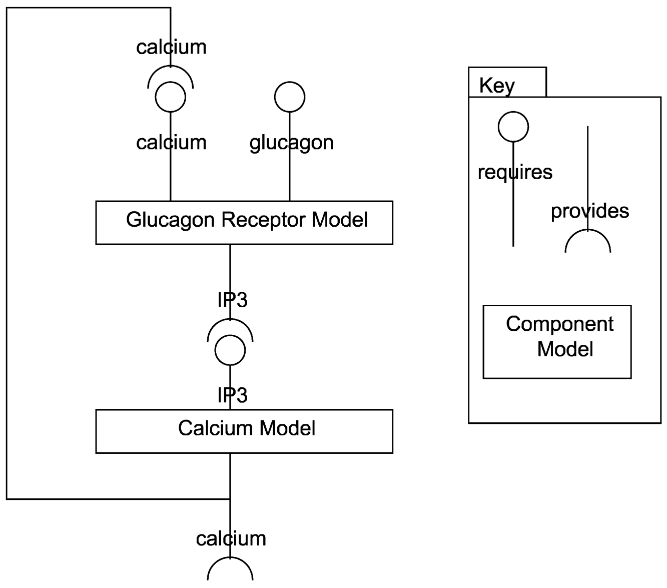


Fig. 7. The two modules that constitute our example model, their outputs and driving functions, and how the models interconnect.

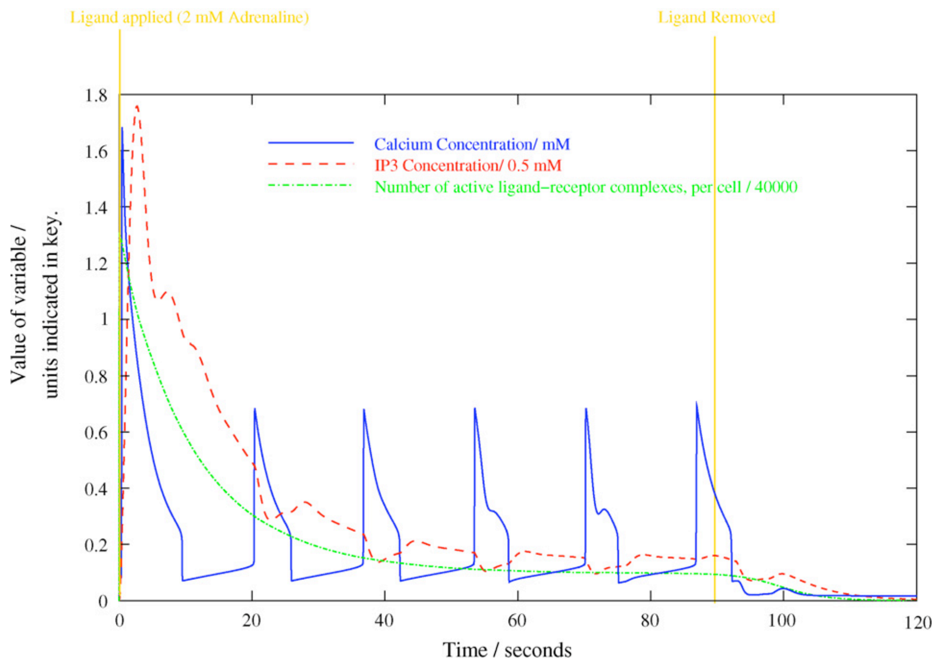


Fig. 8. Results for the example model.

important feature in many signalling pathways; this is an ample of how a module could be re-used in a later model. enomena represented in this model comprise ligand-receptor nding, activation of G-protein, and release of Inositol Trispho-hate (IP3). This model is based on Nauroschat and an der eiden (1997) and Riccobene, Omann, and Linderman (1999), odel of G-protein linked receptor phenomena including desen-isation. To these models we have added the known effect that lcium increases receptor inactivation, as modelled in Kummer, sen, Dixon, Green, and Bornberg (2000). The processes mod-ed are: ligand-receptor binding and dissociation, receptor questration and desequestration (and its dependence on recep-r phosphorylation state), receptor phosphorylation (and its pendence on active G-protein and ligand-binding), G-protein tivation and inactivation (and its dependence on calcium and ospholipase C), and the production of phospholipase C and nce IP3 by active G-protein. These are expressed in Mathemat-a as a series of chemical steps, with the appropriate ODEs being nerated automatically by Mathematica's symbolic engine.

The second module describes the signalling pathway acti-ted by the IP3 released by the first model—the 'cyto-asm' module, which describes the effects of the signal thin the cell. This model of calcium oscillations is built in PPAUT (Ermentrout, 2000) and is based on a model by Hofer 999). It is a simplification of Hofer's model, as discussed Hetherington, Warner, and Seymour (2005). The processes volved include IP3-dependent calcium entry through the cell embrane, calcium- and IP3-dependent release of calcium by e endoplasmic reticulum (ER), and the ER and membrane cal-um pumps.

Thus, the chosen configuration of modules contains a feed-ck loop; the PLC levels provided by the membrane module t as an input for the cytoplasm module. In return, the cal-um levels are provided as an input for the membrane module.

These are the quantities which are reported by each model and passed to the other model by the connector. The next element of the pathway—the action of calcium on glycogen phosphorylase via phosphorylase kinase and phosphorylase phosphatase, is not covered in this two model example, but is covered in detail in our forthcoming larger model.

We present in Fig. 8 some results from this example. We observe calcium oscillations, which occur as calcium moves between the endoplasmic reticulum and cytosol, due to the phenomenon known as calcium-induced-calcium release. However, comparison with the results of the calcium oscillation model without feedback to the receptor Fig. 9 shows that the feedback between the two models transiently alters the shape of the oscillations, and we are preparing a paper on the scientific implications of this finding.

The scientific results obtained are encouraging, but the evidence that our framework is of benefit, the subject matter of this paper, occurs not in the scientific results themselves, which can be recreated with a more traditional approach, but in the

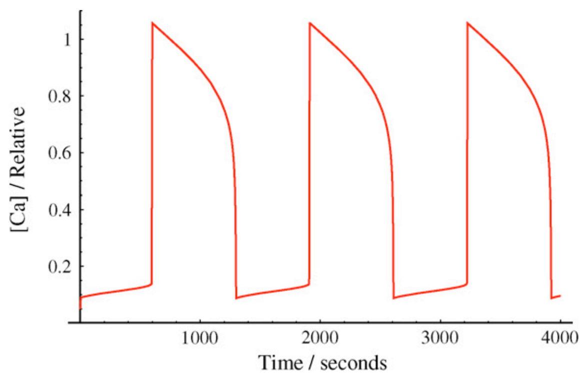


Fig. 9. Typical calcium oscillation results from the simplified calcium oscillations model in the absence of feedback to the receptor module.

prevention of mistakes in the development process. We do not claim that the model we present here could not have been built without our techniques, but rather, that with them, its construction is safer and conclusions based upon it are consequently more robust. In order to provide evidence for this claim, we must contrast the way model information management and model integration work in our system with the way these issues have been dealt with previously, and we do so in the following sections.

### 8.2. *Integrated modelling*

Building a model of two interconnected phenomena by combining existing models implemented in their own environments indeed provided advantages we had anticipated.

It was not necessary to translate the models into a common language, which might have introduced errors. Modellers were able to develop models in the environments they were used to, meaning development and testing were significantly faster. While it was necessary to modify the models slightly to allow for their interaction, this work was done as a seamless development from the original model implementation, in their original context. Stand-alone validation of the modified component models carried over directly to the compound model.

As the two modelling environments had very different typical use patterns and approaches to model specification, parameter definition, and very different APIs, the development of common access forms for the framework was particularly challenging in this case. Our policy of requiring minimal changes of the model files (replacing NDSolve with NDFramework in the Mathematica case and specification of input and output variables in the XPPAUT case) means that individual models may be imported with alacrity. In our example, the equations for the Mathematica model are defined not as ODEs, but as a series of chemical processes, with a Mathematica library providing the conversion using Mathematica's symbolic engine, according to the appropriate kinetics. Thus, we see that by using the Mathematica engine to execute the model definition file as defined by the modeller, we retain the full expressiveness of the Mathematica language. Indeed, zero parameter values will result in Mathematica's symbolic engine compiling away appropriate model terms, with associated efficiency gains.

In addition, the approach allowed distributed development of the test case, with separate developers responsible for each component model. It is clear to us that the many well established advantages of modularity within the software engineering community carry over into biological modelling. However, a more interesting and unexpected advantage of this approach results from the nature of the biological domain—interacting systems acting on different scales and at differing physical locations within the cell. By letting the biological system's own modular nature express itself within the breakdown into modules, each of the software engineering arguments for modularity is made stronger—for example, division of labour for modellers is even more useful, as expertise is more specialised. In addition, the weakly coupled clusters nature of the graph of interactions of a typical biological system is perfectly suited to modularity. In our example, the membrane-phenomena-in-one-model, cytoplasmic-phenomena-in-the-other approach provided an accidental separation of scales which fits well with the waveform relaxation algorithm.

However, there were significant difficulties with building the wrappers which made the implementation of our approach more difficult, the most important of which were undocumented unpredictable behaviour in the Mathematica C++ API and inadequate functionality in the XPPAUT C++ API which necessitated modification of the XPP source code.

### 8.3. *Context management*

Our parameter information, instead of residing haphazardly in a variety of mathematician's notebooks and model files, reside in XML files on a parameter server, which can be easily examined using our tools.

This provides the first clear advantage of our approach, for the provenance data associated with the parameter values is clearly exposed to all members of the project. (Modellers' model files or notebooks can be hostile territory for biological colleagues.) This resulted in an immediate response from biologists regarding which parameter values were trustworthy and which were questionable. Data from certain publication venues or obtained experimentally with older, deprecated experimental techniques or from inappropriate animal or cell models was known to be of low quality. For example, much calcium oscillation data is obtained from experiments in *Xenopus* oocytes, but this data is not always appropriate for liver modelling.

Within our framework, biologists can easily review the parameters and the evidence the modellers have found for them, they can ensure that parameters are "well-audited". The distributed manner in which the parameter database can be accessed and modified (from a variety of physical locations) was particularly important in facilitating collaborative context management. For example, it was suggested that the receptor sequestration form of adaptation might not be important for hepatocytes, so an alternative case with $k_s = 0$ much lower was investigated. Another example is a set of calcium oscillation parameters obtained by our own literature search, independent of the set obtained by matching our simplified model of calcium oscillations to Hofer (1999). Since the "correctness" of these alternative parameter sets is hard to determine support for maintenance of several alternative parameter sets has been extremely useful.

We maintain a record of parameter values used for each run of the orchestrated model, in the form of the "parameter run files" (PRFs) generated by the context database immediately prior to execution of the model. Of course, the existence of the ability to save and load parameter sets is not new, but we emphasise that we have achieved it across multiple modelling environments and languages, each with its own way of handling parameters, and where this separation is unsupported. The existence of the parameter run file and context database encourages separation of two concerns in model development: parameter selection on the one hand, and the development of model equations on the other. Existing biological model definition languages (SBML, CMSL, XPPAUT, etc.) often force these concerns to be interwoven.

Our “interpretation service” enables the association of par-
ular parameter values with the conclusions of model analyses,
rough the production of the automatic model reports. In our
o-model example, the end-to-end information management
eant that within the automated model reports, it was clear to
team members when a parameter had been tuned away from
literature-supported value to obtain interesting results (for
ample, the case of exaggerated calcium feedback shown in
e results figure in Section 8.1.) In other test systems we have
und that automated detection and updating of shared parame-
rs (for example, the calcium pump rate shared between blood
d cell models in the seven-module example Appendix A.2)
rther prevents time-wasting mistakes.

## Conclusions

We have presented a model integration framework for sys-
ms biology, with an architecture based on an orchestrator,
rappers, connectors, and information services. We have built
any software components which together constitute an imple-
entation of this system. By the development of our two-model
ample we have demonstrated some of the advantages of our
proach, which brings well-established benefits of modern soft-
are engineering techniques to systems biology. Our aim is
ultiscale modelling, where we link models based on different
eas of biological expertise. We find that a modular, composi-
onal approach is highly suited to this problem and that support-
g interoperability between modelling environments permits
e composition of models developed by experts with their
vn modelling environment preferences. Since multiscale mod-
ing requires teamwork between modellers and biologists from
ry different areas, communication and information sharing
sues become much more challenging. Sophisticated informa-
on management is thus particularly important in this area.

The framework supports sophisticated queries which will
able one to associate patterns in model results with particular
perimental techniques—in future we expect this data-mining
proach to model analysis to prove fruitful. In our continuing
ork, we are applying our framework to a larger test system—a
mplete model of the glucose homeostasis system. We hope to
d support within our framework for managing the evolution
models—version control for systems biology.

## cknowledgement

We gratefully acknowledge the funding of the United King-
om Department of Trade and Industry (DTI).

## ppendix A. A example model

### 1. Example model equations and typical parameter lues

We present here the equations, definitions of terms, and
fault parameter sets for the two modules.

#### A.1.1. Receptor module

$$\frac{\mathrm{d}R_\mathrm{r}}{\mathrm{d}t} = k_{-1}\mathrm{LR_u} - L(t)k_1 R_\mathrm{r} - k_\mathrm{s} R_\mathrm{r} + k_\mathrm{r} R_\mathrm{s} \tag{A.1}$$

$$\frac{\mathrm{d}R_\mathrm{s}}{\mathrm{d}t} = k_\mathrm{sp}\mathrm{LR_p} + G_i K_{2\mathrm{s}}\mathrm{LR_u} + k_\mathrm{s}(\mathrm{LR_u} + R_\mathrm{r}) - k_\mathrm{r} R_\mathrm{s} \tag{A.2}$$

$$\frac{\mathrm{d}G_i}{\mathrm{d}t} = -G_i K_{23}\mathrm{LR_u} + G_* \left(k_\mathrm{h} + \frac{\mathrm{Ca}[t]k_{\mathrm{Gdeg,Cal}}}{K_{\mathrm{Gdeg,Cal}} + G_*} + \frac{k_{\mathrm{Gdeg,PLC}}\mathrm{PLC}_*}{K_{\mathrm{Gdeg,PLC}} + G_*}\right) \tag{A.3}$$

$$G_* = G_0 - G_i \tag{A.4}$$

$$\frac{\mathrm{d\,LR_p}}{\mathrm{d}t} = -k_\mathrm{sp}\mathrm{LR_p} + \frac{k_\mathrm{p}(1 + (A_0/(1 + B_1 G_*^{-n_1})))}{1 + B_2\mathrm{LR_u}\mathrm{LR_u}^{-n_2}} \tag{A.5}$$

$$R_0 = R_\mathrm{r} + R_\mathrm{s} + \mathrm{LR_u} + \mathrm{LR_p} \tag{A.6}$$

$$\frac{\mathrm{d\,PLC}_*}{\mathrm{d}t} = k_\mathrm{PC} G_* - \frac{k_{\mathrm{PC,deg}}\mathrm{PLC}_*}{k_{\mathrm{PC,deg}} + \mathrm{PLC}_*} \tag{A.7}$$

$$P = k_\mathrm{conv}\mathrm{PLC}_* \tag{A.8}$$

The quantities $R_\mathrm{r}$, $R_\mathrm{s}$, $\mathrm{LR_u}$ and $\mathrm{LR_p}$, are respectively the free receptor, sequestered receptor, ligand-bound receptor, and phosphorylated, ligand-bound receptor. The quantities $G_i$ and $G_*$ are respectively the inactive and active G-proteins. $\mathrm{PLC}_*$ is the active phospholipase *C*, and *P* is inositol trisphosphate. Most quantities are given as numbers-of-molecules per cell—an unfortunate convention as they are extensive quantities which are less likely to transfer between cells, but a convention we have adopted from the papers on which these parts of the model was based. The exceptions are the input functions $L(t)$ and $C(t)$, (respectively, the concentration of the hormone in the blood to which the receptor responds and the concentration of calcium ions in the cytoplasm) *P* and $\mathrm{PLC}_*$, which are defined in micromolar. Time is defined in seconds. There is insufficient space in this methodological paper to go into the detailed assumptions which have been used to obtain these equations, but we present here without further justification the canonical parameter values used in our test case, in the appropriate units as defined above: $k_{-1} = 10^1$, $k_1 = 10^2$, $k_\mathrm{s} = 5.2 \times 10^{-3}$, $k_\mathrm{s}\mathrm{p} = k_\mathrm{s}$, $K_{2\mathrm{s}} = 2.0 \times 10^{-8} k_\mathrm{s}$, $k_\mathrm{r} = 4.0 \times 10^{-3}$, $K_{23} = 1 \times 10^{-7}$, $k_\mathrm{h} = 2.0 \times 10^{-1}$, $k_{\mathrm{Gdeg,Cal}} = 1.47 \times 10^3$, $K_{\mathrm{Gdeg,Cal}} = 3.54 \times 10^1$, $k_{\mathrm{Gdeg,PLC}} = 2.19 \times 10^3$, $K_{\mathrm{Gdeg,PLC}} = 5.7$, $k_\mathrm{p} = 6.5 \times 10^4$, $A_0 = 3.0$, $B_1 = n_1 = 1$, $B_2 = 10^6$, $n_2 = 1$, $R_0 = 5.5 \times 10^4$, $G_0 = 10^5$, $k_\mathrm{PC} = k_{\mathrm{PC,deg}} = 2.82 \times 10^{-1}$, $K_{\mathrm{PC,deg}} = 2.55 \times 10^{-1}$, $k_\mathrm{conv} = 10^2$. The origins of these parameter values will be discussed in detail in a forthcoming paper. Note that the model is defined in Mathematica not in terms of the above algebra, but in the language of chemistry. So that, for example, $A \xrightarrow{A,k,M} B$ represents a Michaelis–Menten enzyme reaction catalysed by *A* with max rate *k* and Michaelis constant *M*, while $A \overset{k_\mathrm{f}k_\mathrm{r}}{\rightleftharpoons} 2B$ represents a pair of reactions with mass-action kinetics in equilibrium.

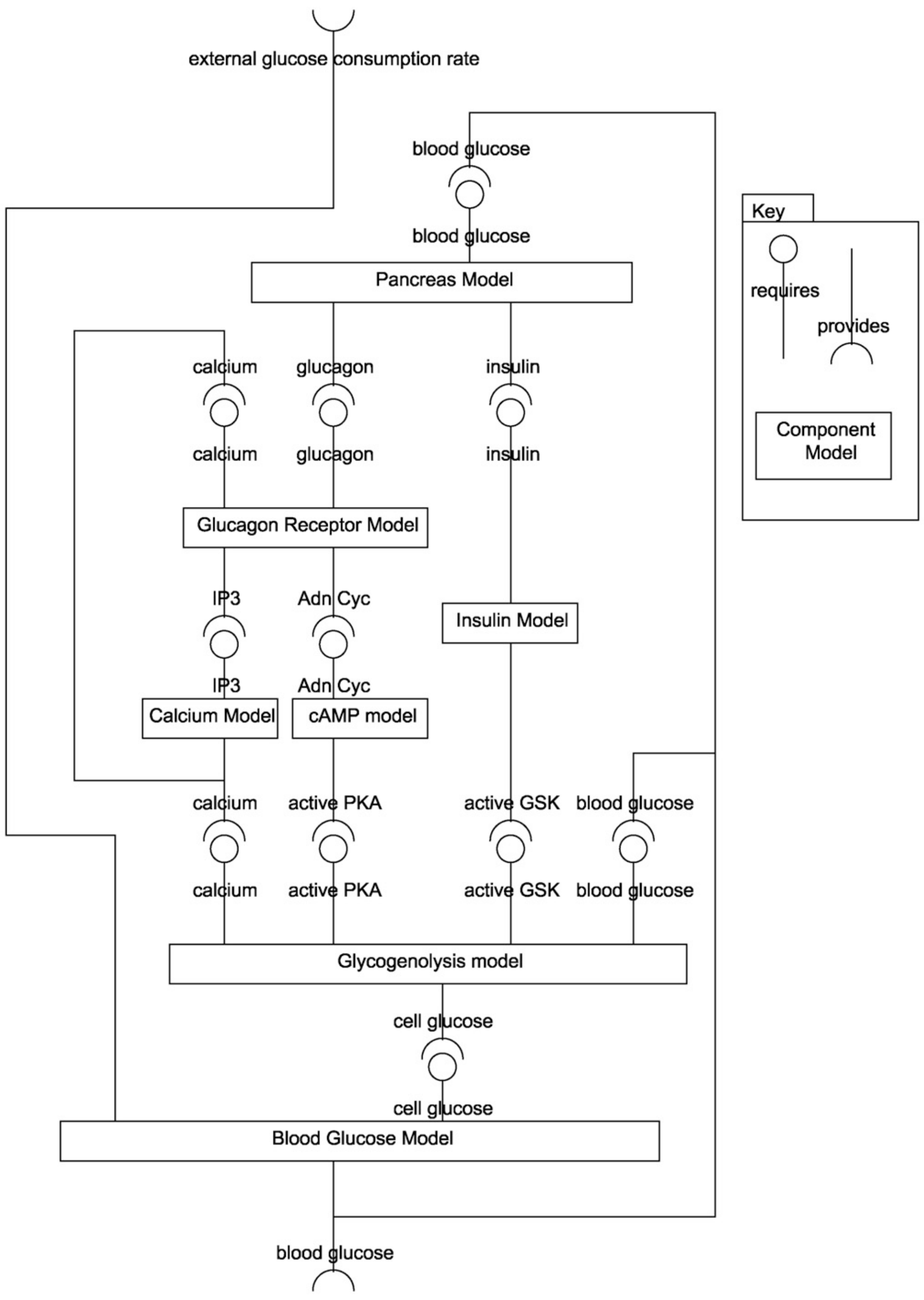


Fig. 10. An example of the kind of larger composite model that can be built with the framework. This model has seven subcomponents, and multiple nested feedback loops. It is a model of glucose homeostasis, in terms of the interaction between the liver and the pancreas, and will be described in a future publication.

### *A.1.2. Calcium module*

We begin our definition of the calcium module by breaking the rate equations for cytoplasmic calcium $C$ and endoplasmic reticulum calcium $E$ into components:

$$\frac{\mathrm{d}C}{\mathrm{d}t} = J_{\mathrm{ER}} + J_{\mathrm{PM}} \quad (A.9)$$

$$\frac{\mathrm{d}E}{\mathrm{d}t} = -\upsilon J_{\mathrm{ER}} \quad (A.10)$$

where $J_{\mathrm{ER}}$ is the net rate of flow of calcium between the ER and the cytosol, and $J_{\mathrm{PM}}$ is the net rate of flow of calcium between the cytosol and the external medium. These are separated into positive and negative parts:

$$J_{\mathrm{X}} = J_{\mathrm{X,in}} - J_{\mathrm{X,out}} \quad (A.11)$$

with X = (ER,PM).

We then use the above notation to define the model:

$$J_{\mathrm{ER,in}} = k_{\mathrm{EC}}(E - C)(l_{\mathrm{EC}} + U(P(t), C)) \quad (A.12)$$

$$J_{\mathrm{ER,out}} = k_{\mathrm{EP}}\Theta_n(C, c_{\mathrm{EP}}) \quad (A.13)$$

$$J_{\mathrm{PM,in}} = S(t)k_{\mathrm{MC}}(l_{\mathrm{MC}} + \Theta_n(P(t), p_{\mathrm{MC}})) \quad (A.14)$$

$$_{\rm M,out} = k_{\rm MP}\Theta_n(C, c_{\rm MP}) \tag{A.15}$$

$$(P, C) = \Theta_n(P, p_{\rm EC})\Theta_n(C, c_{\rm EC,+})(1 - \Theta_n(C, c_{\rm EC,-})) \tag{A.16}$$

ere $P(t)$ is the input function for this module, the concentration inositol trisphosphate. $\Theta_n(a, t) = 1/(1 + (t/a)^n)$, the Hill funcon. The parameter values used are $k_{\rm MC} = 0.08$, $k_{\rm MP} = 0.072$, $_{\rm C} = 2.0$, $k_{\rm EP}18.0$, $c_{\rm EC,+} = 0.26$, $c_{\rm EC,-} = 0.65$, $p_{\rm EC} = 0.45$, $_{\rm P} = 0.12$, $p_{\rm MC} = 4.0$, $c_{\rm MP} = 0.26$, $l_{\rm MC} = 0.05$, $l_{\rm EC} = 0.02$, $v =$ .0, $n = 8$, with variables defined in micromolar and time in conds.

We may now emphasise how the first model takes $C(t)$, one of e variables of the second model, as an input driving function, hile the second model takes $P(t)$, an output of the first model, one of its driving functions. This makes concrete the model terdependence represented in Fig. 7. Note also the time-scale fferences—while receptor activation phenomena respond over nescales as fast as a second calcium oscillations occur on nescales around a minute, justifying on numerical efficiency ounds the use of the waveform relaxation algorithm.

## 2. *Larger example*

We are conducting further tests on a larger example, with ven component models, of a more significant portion of the ucose homeostasis system, including the pancreatic hormones sulin and glucagon. This model displays the ultradian oscillans sometimes observed in glucagon and pancreas (Simon & randenberger, 2002), and will be the subject of a forthcoming ologically focussed publication (Fig. 10).

## ppendix B. Waveform relaxation algorithm

Take some time interval $I = [t_0, T]$ $(T < \infty)$ and let $F$ denote suitable function space consisting of real-valued functions fined on $I$; by suitable we mean at least continuous, and ossibly continuously differentiable. Consider a set of $N$ mods indexed by $i = 1, 2, \ldots, N$ and defined as mappings $M^i$ om product spaces $F^i : \prod_{j=0}^{s_i} F$ to itself; here $s_i$ is the numr of functions $f_j^i$ required to define the $i$th model, and $i = 0$ rresponds to external forcing functions. The mappings $M$ fine the time evolution of the underlying model dynamics; r example in the case of ordinary differential equations they e a time-integral operator. Then $f_j^i = M_j^i[g]$ takes as inputs nctions $g \in \prod_{i=0}^{N} F^i$ and produces as an output a function $= (f_0^i, f_1^i, j_2^i, \ldots, f_{s_i}^i) \in M^i$. The input functions $g^i$ of the model are either given as external driving functions or are tput functions of other models in the model set; the details the input/output structure are defined via the composition atrix $M_{rs}^{pq}$, where $p, q$ index the input and output model and $r$, he input and output function. Thus, $M_{rs}^{pq} = 1$ iff the relevant put is obtained from the relevant output, and 0 otherwise, i.e. $= \sum_{qs} M_{k,s}^{iq} f_s^q$.

We wish to obtain a solution set $\{f^i\}$ which satisfies the consistency equation $f_j^i = M_j^i\left[\sum_{qs} M_{k,s}^{iq} f_s^q\right]$ for all $i, j$ (which is

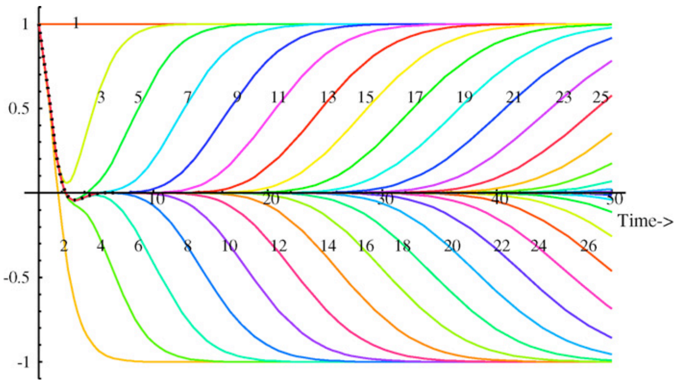


Fig. 11. The waveform relaxation has a tendency to non-uniform convergence, with each successive iteration, (labelled with numbers) leaving the envelope of the correct result after linear increments of time. The correct result curve (shown dashed) rapidly falls to zero, while other curves "fly-off", to values which are attractors for the WR scheme but not for the complete model. Shown here is the variable $x$ from a simple test model $x' = -x - y$, $y' = -y + x$, with each variable's DE treated as one component model.

simply the condition that $f$ is a fixed point of the map $M$ and hence a solution of the model as a dynamical system). The waveform relaxation algorithm takes a set of seed functions $(g_0)_k^i$ and produces new iterates via $(g_{n+1})_j^i = M_j^i\left[\sum_{q,s} M_{ks}^{iq}(g_n)_s^q\right]$ for $n = 0, 1, \ldots$ If this iterative procedure converges, the result will be a consistent solution. We define our convergence test as $\sum_{i,j}(||(f_n)_j^i - (f_{n-1})_j^i||^2/||(f_n)_j^i||^2 + \in) < C$ where $||f|| = \left(\int_{t_0}^{T} f(t)^2\,{\rm d}t\right)^{1/2}$ denotes the $L_2$ norm on $F$ and epsilon is a small quantity defined to stabilise the test when functions are close to zero. In numerical applications, of course, the function space $F$ is replaced by some finite dimensional representation, such as the space of a piecewise linear functions on $I$ equipped with a suitable norm (e.g. Euclidean), but the principle of waveform relaxation remains the same.

The algorithm has a tendency for non-uniform convergence in $t$, such that the "time of fly-off" advances linearly with each successive iteration, see Fig. 11.

## References

Antoniotti, M., Policriti, A., Ugel, N., & Mishra, B. (2003). Model building and model checking for biochemical processes. *Cell Biochemistry and Biophysics*, *38*(3), 271–286.

Ashburner, M., Ball, C., Blake, J., & Botstein, D. (2000). Gene ontology: tool for the unification of biology. *Nature Genetics*, *25*, 25–29.

Bayer, B., & Marquardt, W. (2004). Towards integrated information models for data and documents. *Computers and Chemical Engineering*, *28*(8), 1249–1266.

Belaud, J., Pons, M., & Braunschweig, B. (2002). Open software architecture for process simulation: The current status of cape-open standard. *Computer Aided Chemical Engineering*, *10*, 847–852.

Box, D., Ehnebuske, D., Kakivaya, G., & Layman, A. (2000). *Simple object access protocol (SOAP) 1.1*. W3C Note.

Burrage, K. (1995). *Parallel and sequential methods for ordinary differential equations*. Oxford University Press.

Campbell, A., & Hummel, J. (1998). Dynamic information architecture system: An advanced simulation framework for military and civilian applications. *Society for Computer Simulation International, Simulation Series*, *30*(4), 212–217.

Chen, P. (1976). The entity-relationship model—Toward a unified view of data. *ACM Transactions on Database Systems*, *1*(1), 10–18.

Christensen, E., Curbera, F., & Meredith, G. (2001). *Web services description language (WSDL) 1.1*.

Ermentrout, B. (2000). *XPPAUT*.

Finkelstein, A., Hetherington, J., Li, L., Margoninski, O., Saffrey, P., Seymour, R., et al. (2004). Computational challenges of systems biology. *IEEE Computer*, *37*(5), 26–33.

Hetherington, J., Warner, A., & Seymour, R. M. (2005). Simplification and its consequences in biological modelling: Conclusions from a study of calcium oscillations in hepatocytes. *Journal of the Royal Society: Interface*, (10).

Hofer, T. (1999). Model of intercellular calcium oscillations in hepatocytes: Synchronization of heterogeneous cells. *Biophysical Journal*, *77*(3), 1244–1256.

Hucka, M., Finney, A., Sauro, H., Bolouri, H., & Doyle, J. (2002). The erato systems biology workbench: Enabling interaction and exchange between software tools for systems biology. *Proceedings Pacific Symposium on Biocomputing*.

Hucka, M., Finney, A., Sauro, H., Bolouri, H., & Doyle, J. (2003). The systems biology markup language (SBML): A medium for representation and exchange of biochemical models. *Bioinformatics*, *19*(4), 524–531.

Hunter, P., Robbins, P., & Noble, D. (2002). The iups human physiome project. *Pflugers Archives*, *445*(1), 1–9.

Joshi-Tope, G., Gillespie, M., & Vastrik, I. (2005). Reactome: A knowledgebase of biological pathways. *Nucleic Acids Research*, *33*(Database Issue (D)), 428–432.

Kanehisa, M., Goto, S., Ogata, H., Sato, K., & Fujibuchi, W. (2000). Kegg: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Research*, *28*(1), 27–30.

Kuhl, F., Weatherly, R., & Dahmann, J. (2000). *Creating computer simulation systems: An introduction to the high level architecture*. Prentice Hall PTR.

Kumar, S., & Feidler, J. (2003). Biospice: A computational infrastructure for integrative biology. *Omics A Journal of Integrative Biology*, *7*(3), 225.

Kummer, U., Olsen, L., Dixon, C., Green, A., & Bornberg, E. (2000). Switching from simple to complex oscillations in calcium signaling. *Biophysical Journal*, *79*(3), 1188–1195.

Lloyd, C., Halstead, M., Nielsen, P., & Bullivant, D. (2004). CellML: Its future, present and past. *Progress in Biophysics and Molecular Biology*, *85*(2/3), 433–450.

Loew, L., & Schaff, J. (2001). The virtual cell: A software environment for computational cell biology. *Trends in Biotechnology*, *19*(10), 401–406.

Margoninski, O., Saffrey, P., Hetherington, J., Finkelstein, A., & Warner, A. (in press). A specification language and a framework for the execution of composite models in systems biology. *LNCS Transactions of Computational Systems Biology*, VII (LNBI 4230).

Meier, W. (2002). Exist: An open source native XML database. *Web, Web-Services, and Database Systems*.

Nauroschat, J., & an der Heiden, U. (1997). A theoretical approach to G-protein modulation of cellular responsiveness. *Journal of Mathematical Biology*, *35*, 609–627.

Noble, D. (2002). Modeling the heart-from genes to cells to the whole organ. *Science*, *5560*, 1678–1682.

Novére, N. L., Finney, A., Hucka, M., & Bhalla, U. (2005). Minimum information requested in the annotation of biochemical models (Miriam). *Nature Biotechnology*.

Riccobene, T. A., Omann, G. M., & Linderman, J. J. (1999). Modeling activation and desensitization of G-protein coupled receptors provides insight into ligand efficiency. *Journal of Theoretical Biology*, *200*(2), 207–222.

Saffrey, P., Margoninski, O., Hetherington, J., Varela-Rey, M., Yamaji, S., & Finkelstein, A., et al. (2006). End-to-end information management for systems biology. *LNCS Transactions of Computational Systems Biology*, submitted for publication.

Schomburg, I., Chang, A., Hofmann, O., Ebeling, C., & Schomburg, D. (2002). Brenda: a resource for enzyme data and metabolic information. *Trends in Biochemical Sciences*, *27*(1), 54–56.

Schuler, G., Epstein, J., Ohkawa, H., & Kans, J. (1996). Entrez: Molecular biology database and retrieval system. *Methods in Enzymology*, *266*, 141–162.

Sharma, N., Ierapetritou, M., & Yarmush, M. (2005). Novel quantitative tools for engineering analysis of hepatocyte cultures in bioartificial liver systems. *Biotechnology and Bioengineering*, *92*(3), 321–335.

Simon, C., & Brandenberger, G. (2002). Ultradian oscillations of insulin secretion in humans. *Diabetes*, *51*(Suppl. 1).

Takahashi, K., Kaizu, K., Hu, B., & Tomita, M. (2004). A multi-algorithm, multi-timescale method for cell simulation. *Bioinformatics*, *20*(4).

The EMP Project. (1999). *The EMP project*. http://www.empproject.com/about/.

The UCL Beacon Project. (2002–2007). *The UCL Beacon Project*. http://grid.ucl.ac.uk/biobeacon/.

Tomita, M., Hashimoto, K., Takahashi, K., & Shimizu, T. (1999). E-cell: Software environment for whole-cell simulation. *Bioinformatics*, *15*(1), 72–84.

Vazquez-Roman, R., King, J., & Banares-Alcantara, R. (1996). Kbmoss: A process engineering modelling support system. *Computers and Chemical Engineering*, *20*, 309–314.

Vinoski, S. (1997). Corba: Integrating diverse applications within distributed heterogeneous environments. *IEEE Communications Magazine*, *35*(2), 46–55.

Winer, D. (1998–2003). *XML-RPC specification*.

Zdobnov, E., Lopez, R., Apweiler, R., & Etzold, T. (2002). The EBI srs server-recent developments. *Bioinformatics*, *18*(2), 368–373.