BASIS: an internet resource for network modelling

Colin S. Gillespie [ac], **Darren J. Wilkinson** [ac], **Daryl P. Shanley** [bc], **Carole J. Proctor** [bc], **Richard J. Boys** [ac], **Thomas B.L. Kirkwood** [bc]

[a]School of Mathematics & Statistics, Newcastle University, UK,
[b]Institute for Ageing and Health, and SCMS - Gerontology, Newcastle University, UK.
[c]Centre for Integrated Systems Biology of Ageing and Nutrition, Newcastle University, UK

Summary

There is a growing realisation that complex biological processes cannot be understood through the application of ever more reductionist experimental programs alone. Recognising this, we have a constructed a flexible web-service based modelling system called BASIS (Biology of Ageing e-Science Integration and Simulation), which facilitates model construction and development. In particular it allows users to store, share and simulate their models. The system is accessed through web-services using any language (e.g. Python or Java) or under any operating system (e.g. Linux or Windows).

1 Introduction

As a result of recent advances in experimental techniques, biology has become much more of an informational science. The capacity to answer questions ranging from cell and molecular function through to the population requires an increasing ability to acquire, store, and manipulate large volumes of raw data in a flexible, efficient manner. Moreover, there is a growing realization that complex biological processes cannot be understood through the application of ever-more reductionist experimental programs.

There is a developing perception that mathematical modelling provide some of the necessary tools required to understand this mass of biological data. Indeed, there are distinct advantages of modelling a biological process with the rigour needed to build a mathematical model. First, when constructing a model, gaps in current knowledge are highlighted[14, 25]. Even the very process of model specification will highlight important unknowns. Second, when building a model, verbal hypotheses are made specific and conceptually rigorous[2, 6]. Third, models can yield quantitative as well as qualitative predictions[3, 16].

However, despite the important contributions that are made by models, they are often limited in their usefulness as they tend not to be easily accessible by those inexperienced in modelling. For these reasons, we began a project to create a web-service based modelling system known as the BASIS (Biology of Ageing e-Science Integration and Simulation) system[13]. The primary objective of BASIS is to help advance the understanding of the complex biology of ageing, where many different mechanisms act and interact at a range of different levels. However, the capabilities of BASIS are generic to a wide range of other biological systems. Our system aims to make both existing and new models accessible to the research community in a way that users can adapt models and run simulations themselves. We have adopted the Systems Biology Markup Language (SBML) which is a computer-readable format for representing models of biochemical reaction networks (see section 2.1 for further details). Currently BASIS is unique

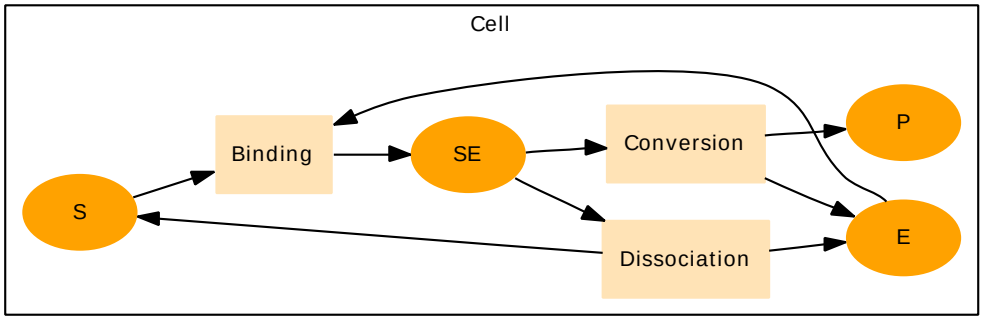

Figure 1: *A simple biochemical network.*

in that it allows users to interact with advanced modelling facilities through a web service API. Furthermore, parts of the system can be accessed through a web-browser, or downloaded and implemented outside of BASIS.

The BASIS project is supported by a team from a wide range of disciplines including biological sciences, mathematical and statistical sciences, and computer science. The aim is to bring the experimental scientists and mathematical modellers into close collaboration, and is already being realised (see [21] for an example). By sharing and integrating models and data, advances have been made in our understanding of ageing. BASIS also provides open source downloadable tools in addition to a comprehensive online simulation system and modelling environment ([7] provides details).

2 The system

2.1 Systems biology markup language

In general, the model in the BASIS system can be envisaged as networks of individual biochemical mechanisms, represented by a system of chemical equations, quantified by substrate and product concentrations and the associated reaction rates. The models can easily be represented in a standard biochemical diagram (see Figure 1). They are described using SBML[10], which is essentially an eXtensible Markup Language (XML) encoding of the reaction, species and compartment lists, together with the additional information required for quantitative modelling and simulation. SBML is quickly becoming the *lingua franca* for the development and sharing of models of biochemical networks[1, 5, 17]. The current version of SBML allows us to encode and distribute a large class of biochemical network models easily. However there is still a great deal to add to the SBML specification, for example it is currently difficult to represent tissues composed of detailed cellular models.

2.2 Web-services

A web-service is a software system designed to facilitate machine to machine interaction over a network. Messages transmitted between web-services are encoding using SOAP[28]. These

messages are then usually transmitted using HTTP over port 80. Since the web-service communicates through the standard web port, this simplifies the configuration of many web-services since messages can easily pass through fire-walls. Web-service interfaces are described in the Web Services Description Language (WSDL)[29]. These files describe how web-services should communicate with each other, and what arguments should be sent/received. SOAP facilitates the Service-Oriented Architectural(SOA) pattern, that is providing loosely coupled and highly inter-operable applications. Essentially, since web-services communicate using the agreed SOAP message system, this enables inter-operability between different languages (e.g. Java and Python) or different operating systems (e.g. Linux and Windows).

2.3 The BASIS system architecture

The BASIS system of model definition, simulation and visualisation is exposed through several web-services that are served via Apache (see Figure 2). To provide an initial entry point to the BASIS system we have constructed a user-friendly web portal for simple model adjustment and to demonstrate the range of services available (see section 3.1 for further details). The web-services interact with a postgresql database[19] and the job scheduler, Condor[27]. All details of the underlying technology are hidden from the user.

To interact with the services that BASIS provides, a user must first register (this is simply to allow the user to retrieve their models and simulation results). To register with BASIS, a user can either visit the web-site (https://www.basis.ncl.ac.uk/basis/) or use the web-service *createUser*. When registering, a valid email address is required, to discourage potential abuses of the system.

The majority of the web-services provided by BASIS require a session id as an argument. A session id is obtained with the *getSessionId* web-service. As each user logs on, a unique session id is generated, with each being deleted after one hour of inactivity. In theory users never deal with the session id, rather the software tool should provide a user-friendly front-end. For example, the BASIS web-site stores the session id on the client's web browser as a cookie. Using the session id, the user can then interact with the BASIS system.

Initially when a user places a SBML model into the BASIS system, the model is designated private and is only accessible by that user. A user can make their model public (after publication say). Once a model is public, it can not be deleted.

Every model entered into the BASIS system is assigned a model URN to uniquely identify the model. The model URN has the form *urn:basis.ncl:model:#1* where *#1* is an integer.

When a model has been placed into the BASIS system, a user can simulate it using the Gillespie algorithm. We use a stochastic simulator, which is built using the efficient GNU scientific libraries and libSBML. It currently supports local and global parameters, events (without delays), distributions and assignment rules. The simulator can be downloaded separately and installed on local machines if required[7]. One of the novel features of BASIS is that when a model has been simulated using the BASIS system, the results are automatically associated with that particular model. Therefore, when a model is public, all its associated simulation data is also public. This allows a pooling of results, i.e. a user can access simulation results from other users (provided the model is public).

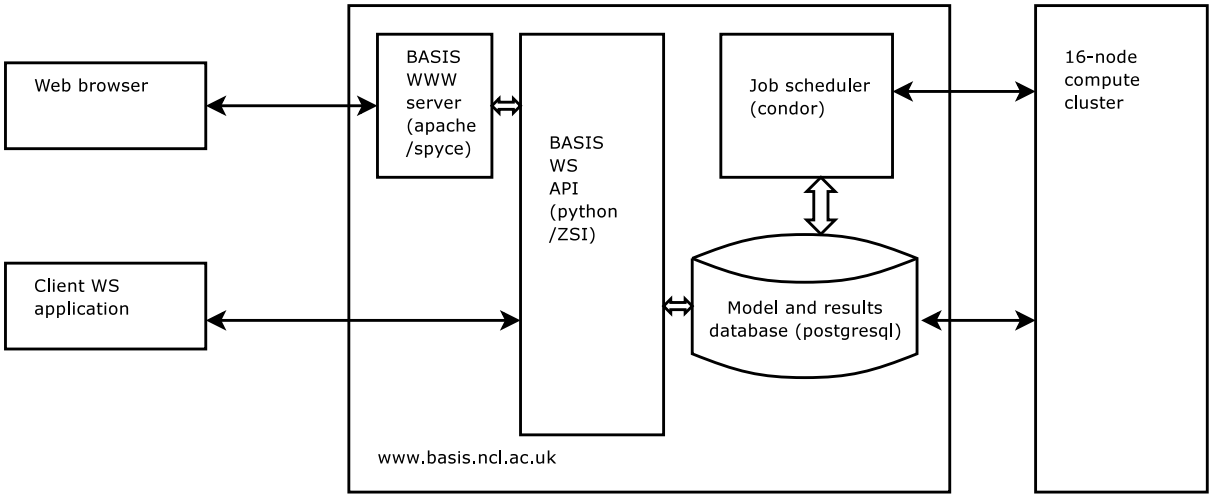

Figure 2: *BASIS Architecture*

Access to the simulator on the BASIS system is again via the web-service interface. All simulations are given a urn of the form:

urn:basis.ncl:model:#1:simulation:#2-#3:#3

where #1 is an integer and refers to the model being simulated, #2 is the simulation time, #3 is the number of time-points to store, #4 is an integer and ensures that the urn is unique and #5 is an optional number referring to a specific simulation. So for example

urn:basis.ncl:model:401:simulation:100000-1000:418

refers to the simulation of model *urn:basis.ncl:model:401*, which has been simulated from $t = 0$ to 100,000 and outputted at 1,000 iterations.

A user can submit as many simulations as they like. However, at most two of their simulations will be running at any one time. The other simulations are placed in a queue. In the near future we plan to join the Newcastle University Grid. This Grid will essentially link all the unused computing resources in Newcastle University and allow jobs to be scheduled when free resource time is available.

2.4 Web-service methods

The web-services provided by BASIS can be roughly split into three areas: user, model and simulation services. The user services deal with the mundane but important matters, such as logging on/off to BASIS, changing passwords and retrieving a lost password.

The model services allow users to obtain, submit and view SBML models. A few example services are:

- *putSBML(sessionId, sbml)*

  - Puts an sbml model into your private space. The model must be valid SBML;
  - Returns 1 if successful.

- *delSBML(sessionId, modelUrn)*
  - Deletes the model from the database. If the model is public, then the model will hidden from view, but still accessible to those that know the modelUrn;
  - All simulation data associated with the model is also deleted;
  - Returns 1 if successful.

- *getMyModelInfo(sessionId)*
  - Returns information regarding a user's private model space.

The simulation services deal with submitting a model and retrieving the results. When a model is private, then all the associated results are private. However, when a model has been made public, the results are automatically made public. This means that simulations of a particular public model are also public, allowing users to efficiently combine stochastic simulations. Example services are:

- *simulate(sessionId, modelUrn, runName, maxTime, no_of_sims, no_of_iters)*
  - Sets off a stochastic simulation on the BASIS system;
  - The service returns a simulation urn.

- killSimulation(sessionId,simulationUrn)
  - Stops a simulation but does not delete the data;
  - Returns 1 if successful.

The WSDL file describing all the BASIS web-services is available from http://www.basis.ncl.ac.uk/basis.wsdl.

3 BASIS in action

3.1 Model building and the BASIS web resources

To provide an initial entry point into the BASIS system we have constructed a user-friendly web portal interface for simple model adjustment and to demonstrate the range of services available (see https://www.basis.ncl.ac.uk/basis/). When the user has logged on to BASIS, they are initially presented with their private model space. From this page, they can add, delete, copy or simulate an SBML model. Additionally, we have a constructed an online model creation facility. We do not foresee that users would create large, complicated models online. Instead we expect users would come to BASIS with a pre-existing model. Their model could be constructed using one of the many SBML tools already available (e.g. [24, 26, 23, 4]) or taken from a SBML model-repository (see [18, 15])

Rather they can make simple adjustments to their model before simulation. One unexpected benefit of having a web-site capable of altering models, is that it is now easier to interact with biologists when model building, since the web-site will run on the majority of current web-browsers.

3.2 Calling the web-services

In this section a simple use-case for the BASIS web-services will be presented and analysed. Here a model, `mymodel.mod`, encoded for discrete stochastic simulation in SBML-shorthand[30], will be assumed to reside on a client's local PC. Using the BASIS web-services, this model will be translated into full SBML, validated, visualised, and submitted to the BASIS model database. Subsequently, a batch of simulation jobs will be requested, and results will be downloaded to the client's local PC for detailed analysis. By their very nature, web-services are language independent. However, for concreteness, the process will be illustrated using Python[22]. First some relevant python modules need to be imported, and to simplify subsequent code some appropriate variables are set.

```
import os, sys, SOAPpy, base64, time
file = 'mymodel.mod'
wsproxy = 'http://www.basis.ncl.ac.uk/'
wsproxy += 'web-services/sbml.py'
bwsproxy = 'https://www.basis.ncl.ac.uk/'
bwsproxy += 'web-services/dbServer.py'
uname, passwd = 'XXXX', '*****'
```

The variables `uname` and `passwd` should be set to a valid BASIS username and password pair. Note that the generic SBML web-services are un-encrytped and are sent via the default HTTP port (80), whereas the BASIS-specific web-services are encrytped, and go via the default HTTPS port (443). Next the SBML-shorthand model is read into a python string.

```
s = open(file,'r')
modstring = s.read()
s.close()
```

Some generic SBML web-services are now used to convert the SBML-shorthand model to SBML, validate the model and then produce a graphical representation of the model for local display.

```
ws = SOAPpy.SOAPProxy(wsproxy)
sbml = ws.mod2sbml(modstring)
print ws.validate(sbml)
fp = os.popen("xv -", "w")
ws_model = ws.visualiseModel(sbml)
fp.write(base64.decodestring(ws_model))
fp.close()
```

Note that the above code assumes a UNIX-like environment and the existence of an image-viewer called `xv`. It can be easily modified for other platforms. Now some BASIS-specific web-services are used to upload the model to the BASIS system and record the URN assigned to the model by BASIS.

```
bws = SOAPpy.SOAPProxy(bwsproxy)
sId = bws.getSessionId(uname, passwd)
modURN = bws.putSBML(sId, sbml)
```

The variable `modURN` now contains the URN assigned to this model by the BASIS system, and can be used in subsequent web-service calls for identifying the model. For example, suppose now that five simulated realisations of the process are required over a period of 100 seconds, and that the state of the process is required at 1000 time points (every 0.1 seconds). This can be requested with the following call.

```
simURN = bws.simulate(sId,modURN,"test runs",str(100),5,1000)
```

It is important to note that this call returns as soon as the simulation request has been processed by BASIS. It does not wait until the simulation job has completed. Waiting for job completion requires some sort of notification mechanism. Unfortunately web-services standards relating to notification are still in a state of flux. So while it is likely that future versions of the web-services interface will support more sophisticated notification mechanisms, the current implementation requires a simple "polling" system to be used to test for job completion. The following piece of python code illustrates how this can be achieved.

```
status = bws.getMySimulationGroupInfo(sId, simURN)

while (status['jobStatus'] != "f"):
    print status['jobStatus'], status['simulationsCompleted']
    print "Waiting..."
    time.sleep(10)
    status = bws.getMySimulationGroupInfo(sId, simURN)
print "Job finished!"
```

Note that data is made available as soon as it has been simulated. So in this example, there is no requirement to wait for all 5 jobs to finished before analysing the results of (say) the first run. It should be clear how to modify the above code to return once the first run is complete, if this is what is required. Once the simulation has completed, the data is available for downloading and subsequent analysis. For brevity, the following code snippet simply downloads the data for each model species for each of the 5 independent runs and prints the data to the console. However, it should be clear how to modify the code to do something more useful with the simulation results.

```
species = bws.getMySpeciesInfo(sId,simURN)['speciesName']
for specie in species:
    print specie
```

```
for i in range(1,6):
    print i
    print bws.getMySpecieData(\
      sId, simURN+':'+str(i), specie)['specieValue']
```

The example considered in this section is useful for introducing the essential concepts associated with using the web-services interface, but it is important to understand that more sophisticated interactions are possible. For example, it is very straightforward to use the interface to carry out a model "parameter scan", where many simulation jobs are run with varying model parameter values in order to conduct a sensitivity analysis. The interface can also be used in the context of parameter inference, where jobs are run with varying parameter values in an attempt to find parameters which cause the model to behave most like available experimental data[12]. Indeed this is the whole point of providing a web-services interface. The flexibility of a programmable interface means that users are free to use the system in ways not necessarily envisaged by the service providers.

4 Associated tools and resources

There are numerous tools that have been developed as part of the process of building the BASIS system. Many of these are generic, and have been released as stand-alone tools for the systems biology community.

1. A Python library (pysbml). This library provides a console based modelling system in Python, a tool for the visualisation of an SBML model (see Figure 1), and a tool for converting an SBML model to an HTML file. More information on both installing and using pysbml can be found in the documentation, available on-line at the BASIS website.

2. A stochastic simulator written in ANSI C (gillespie2). The algorithm executes the standard Gillespie algorithm[9]. A swig interface has also been provided to allow the simulator to be imported into Python.

3. SBML-shorthand provides a shorthand notation for SBML that is much easier for humans to read and write than full SBML. The full specification for SBML-shorthand and a conversion tool is available from the BASIS website.

4. Additionally, a variety of tools have been exposed as web-services, these include model visualisation (see Figure 1), validation and conversion to an HTML document for display within a web browser. The WSDL file for these services can be found at http://www.basis.ncl.ac.uk/sbml.wsdl.

5 Conclusion

The BASIS system is a robust, capable and extensible modelling environment with facilities for remote storage and simulation of models represented in SBML developed for the research into ageing community. Many users are currently benefiting from the resources we provide such

as an actively maintained database of models and simulation results, stochastic simulators and compute power to release users from tying up their own machines. BASIS is built with web service technology and is also available to use via a web interface at www.basis.ncl.ac.uk or for more flexible interaction via our web services. We are currently focusing on a number of developments that will further the value of BASIS:

1. Simulators. The stochastic simulator has recently been extended to include a suite of random distributions which can be accessed by using the CSymbol extension in SBML[8]. The development is essential for modelling telomere shortening[20] but the advantages extend beyond models of ageing and we are pushing for their inclusion within the SBML standard. We will soon make available a deterministic simulator on the BASIS system and we are investigating the inclusion of hybrid simulators as a means to isolate parts of a large model where stochastic simulation is essential.

2. Web-service technology. Currently, our security is managed using session ids and passing SOAP messages over a secure socket. In the near future we intend to implement the WS-Security framework.

3. A BASIS Client. One of the clear advantages of using web service technology is that the user is free to custom-build a client to interact with the system. We have provided a web interface which is suitable for minor changes to existing models but not really suitable for large model development. We have however developed many other tools which connect to our web services and have been packaged as pySBML[7]. Work is now under way to provide a fully featured downloadable GUI that embeds these tools within a fully featured interface.

4. Building large models.

   (a) It is highly desirable to build individual models to address specific biological problems and which can exploit biological modularity. Much can be gained however in combining these models. In principle this is possible as all models in BASIS are represented as SBML, but there are a number of problems that must be overcome. The most obvious is to use a common naming system across models (we are working on a basic ontology for models relevant to ageing). A conceptually more difficult problem is to link models not just through common molecular species but by overloading whole processes with an alternative representation. We are working towards automating this process.

   (b) Multiple instances of component sub models. This is important in modelling a population of individually encoded mitochondria within a cell, or cells within in a tissue. There are several proposed SBML extensions that attempt to address this issue, but none is standard or widely implemented.

We envisage BASIS to be a hub where modellers can not only store their own models and organise their development but also to further their research value by facilitating model sharing and integration. BASIS is a valuable addition to the growing range of modelling services which includes both web-based tools (see [24, 26]) and downloadable software (see [23, 4]). However, the true power of BASIS will only be realised once it is linked to other web-service enabled tools such as CaliBayes (www.calibayes.ncl.ac.uk) and KEGG[11], will provide a powerful

system of interest to both theoretical and experimental biologist alike. The fundamental problem is that researchers are increasingly having to grapple with complex working models whose behaviour is difficult to understand from intuition alone. Experimental work to understand these complex systems must be complemented with *in silico* experimentation. Furthermore the increasing cost of experimentation demands that experimental design needs to be optimised. Enabling technologies such as BASIS are essential to overcome such issues. They are also needed to provide an environment where effective collaboration can take place, a necessary requirement for realising the potential of systems biology.

Acknowledgement

This work was funded by BBSRC, MRC, DTI and Unilever plc.

References

[1] N. A. Allen, L. Calzone, K. C. Chen, A. Ciliberto, N. Ramakrishnan, C. A. Shaffer, J.C. Sible, J. J. Tyson, M. T. Vass, L. T. Watson, and J. W. Zwolak. Modeling regulatory networks at Virginia Tech. *OMICS, A Journal of Integrative Biology*, 7:285–299, 2003.

[2] D. Battogtokh and J. J. Tyson. Bifurcation analysis of a model of the budding yeast cell cycle. *Chaos*, 14:653–661, 2004.

[3] K. C. Chen, L. Calzone, A. Csikasz-Nagy, F. R. Cross, B. Novak, and J. J. Tyson. Integrative analysis of cell cycle control in budding yeast. *Molecular Biology of the Cell*, 15:3841–3862, 2004.

[4] P. Dhar, T. C. Meng, S. Somani, L. Ye, A. Sairam, M. Chitre, Z. Hao, and K. Sakharkar. Cellware - a multi-algorithmic software for computational systems biology. *Bioinformatics*, 20:1319–1321, 2004.

[5] A. Funahashi, N. Tanimura, M. Morohasi, and H. Kitano. CellDesigner: a process diagram editor for gene-regulartory and biochemical networks. *Biosilico*, 1:159–162, 2003.

[6] C. S. Gillespie, C. J. Proctor, D. P. Shanley, D. J. Wilkinson, R. J. Boys, and T. B. L. Kirkwood. A mathematical model of ageing in yeast. *Journal of Theoretical Biology*, 229:189–196, 2004.

[7] C. S. Gillespie, D. P. Shanley, D. J. Wilkinson, R. J. Boys, C. J. Proctor, and T. B. L. Kirkwood. Tools for the sbml community. *Bioinformatics*, 22:628–629, 2006.

[8] C. S. Gillespie, D. J. Wilkinson, R. J. Boys, C. J. Proctor, D. P. Shanley, and T.B.L Kirkwood. *Systems Biology Markup Language (SBML) Level 3. Proposal: Distributions within MathML.*, 2005.

[9] D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, 81:2340–2361, 1977.

[10] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, A. P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, A. A. Cuellar, S. Dronov, E. D. Gilles, M. Ginkel, V. Gor, I. I. Goryanin, W. J. Hedley, T. C. Hodgman, J. H. Hofmeyr, P. J. Hunter, N. S. Juty, J. L. Kasberger, A. Kremling, U. Kummer, N. Le Novre, L. M. Loew, D. Lucio, P. Mendes, E. Minch, E. D. Mjolsness, Y. Nakayama, M. R. Nelson, P. F. Nielsen, T. Sakurada, J. C. Schaff, B. E. Shapiro, T. S. Shimizu, H. D. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, and J. and Wang. The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19:524–531, 2003.

[11] M. Kanehisa, S. Goto, S. Kawashima, and A. Nakaya. The kegg databases at genomenet. *Nucleic Acids Research*, 30:42–46, 2002.

[12] T. B. L. Kirkwood, R. J. Boys, C. S. Gillespie, C. J. Proctor, D. P. Shanley, and D. J. Wilkinson. Computer modeling in the study of aging. In S.N. Austad and E.J. Masoro, editors, *Handbook of the Biology of Aging*, pages 334–357. Academic Press, 2005.

[13] T. B. L. Kirkwood, R. J. Boys, C. S. Gillespie, C. J. Proctor, D.P. Shanley, and D. J. Wilkinson. Towards an e-biology of ageing: integrating theory and data. *Nature Reviews Molecular Cell Biology*, 4:243–249, 2003.

[14] A. Kowald and T. B. L. Kirkwood. A network theory of ageing: the interactions of defective mitochondria, aberrant proteins, free radicals and scavengers in the ageing process. *Mutation Research*, 316:209–236, 1996.

[15] Nicolas Le Novere, Benjamin Bornstein, Alexander Broicher, Melanie Courtot, Marco Donizelli, Harish Dharuri, Lu Li, Herbert Sauro, Maria Schilstra, Bruce Shapiro, Jacky L. Snoep, and Michael Hucka. BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic Acids Research*, 34(suppl_1):D689–691, 2006.

[16] D. E. Nelson, A. E. Ihekwaba, M. Elliott, J. R. Johnson, C. A. Gibney, B. E. Foreman, G. Nelson, V. See, C. A. Horton, D. G. Spiller, S. W. Edwards, H. P. McDowell, J. F. Unitt, E. Sullivan, R. Grimley, N. Benson, D. Broomhead, D. B. Kell, and M. R. White. Oscillations in nf-κb signaling control the dynamics of gene expression. *Science*, 306:704–708, 2004.

[17] B. G. Olivier, J. M. Rohwer, and J.-H. S. Hofmeyr. Modelling cellular systems with PySCeS. *Bioinformatics*, 21:560–561, 2005.

[18] B. G. Olivier and J. L. Snoep. Web-based kinetic modelling using JWS online. *Bioinformatics*, 20:2143–2144, 2004.

[19] PostgreSQL. http://www.postgresql.org.

[20] C. J. Proctor and T. B. L. Kirkwood. Modelling telomere shortening and the role of oxidative stress. *Mechanisms of Ageing and Development*, 123:351–363, 2002.

[21] C. J. Proctor, C. Soti, R. J. Boys, C. S. Gillespie, D. P. Shanley, D. J. Wilkinson, and T. B. L. Kirkwood. Modelling the actions of chaperones and their role in ageing. *Mechanisms of Ageing and Development*, 126:119–131, 2005.

[22] Python. http://www.python.org.

[23] S. Ramsey, D. Orrell, and H. Bolouri. Dizzy: Stochastic simulation of large-scale genetic regulatory networks. *Journal of Bioinformatics and Computational Biology*, 3:361–363, 2005.

[24] B. M. Slepchenko, J. C. Schaff, I. Macara, and L. M. Loew. Quantitative cell biology with the virtual cell. *Trends In Cell Biology*, 13:570–576, 2003.

[25] P. D. Sozou and T. B. L. Kirkwood. A stochastic model of cell replicative senescence based on telomere shortening, oxidative stress, and somatic mutations in nuclear and mitochondrial dna. *Journal of Theoretical Biology*, 213:573–586, 2001.

[26] K. Takahashi, N. Ishikawa, Y. Sadamoto, H. Sasamoto, S. Ohta, A. Shiozawa, F. Miyoshi, Y. Naito, Y. Nakayama, and M. Tomita. E-cell 2: Multi-platform E-Cell simulation system. *Bioinformatics*, 19:1727–1729, 2003.

[27] T. Thain, T. Tannenbaum, and M. Livny. Condor and the grid. In Fran Berman, Geoffrey Fox, and Tony Hey, editors, *Grid Computing: Making the Global Infrastructure a Reality*. John Wiley & Sons Inc., December 2002.

[28] W3C: SOAP Version 1.2. http://www.w3.org/tr/soap12/.

[29] W3C: Web Services Description Language (WSDL) 1.1. http://www.w3.org/tr/2001/note-wsdl-20010315.

[30] D. J. Wilkinson. *Stochastic modelling for systems biology*. Chapman & Hall/CRC Press, 2006.