Robust Workflows for Science and Engineering

§ David Abramson, Blair Bethwaite, Colin Enticott, Slavisa Garic, Tom Peachey
† Anushka Michailova, Saleh Amirriazi, Ramya Chitters

§ Faculty of Information Technology,
Monash University,
Clayton, 3800, Victoria, Australia

† Department of Bioengineering,
University of California San Diego, and San Diego Supercomputer Centre
9500 Gilman Drive, La Jolla, USA

ABSTRACT

Scientific workflow tools allow users to specify complex computational experiments and provide a good framework for robust science and engineering. Workflows consist of pipelines of tasks that can be used to explore the behaviour of some system, involving computations that are either performed locally or on remote computers. Robust scientific methods require the exploration of the parameter space of a system (some of which can be run in parallel on distributed resources), and may involve complete state space exploration, experimental design or numerical optimization techniques. Whilst workflow engines provide an overall framework, they have not been developed with these concepts in mind, and in general, don't provide the necessary components to implement robust workflows.

In this paper we discuss Nimrod/K - a set of add in components and a new run time machine for a general workflow engine, Kepler. Nimrod/K provides an execution architecture based on the tagged dataflow concepts developed in 1980's for highly parallel machines. This is embodied in a new Kepler 'Director' that orchestrates the execution on clusters, Grids and Clouds using many-task computing. Nimrod/K also provides a set of 'Actors' that facilitate the various modes of parameter exploration discussed above. We demonstrate the power of Nimrod/K to solve real problems in cardiac science.

1 INTRODUCTION

Scientific workflows have emerged as a powerful technique for specifying and executing complex in-silico experiments. Whilst there are many engines available, almost all share the ability to script data manipulation and computational pipelines that invoke a series of steps, often utilizing distributed resources. Pipeline steps can range from those that perform simple computations locally, to accessing external instruments and databases, through to significant simulations on remote parallel machines. A good review of workflow engines can be found in [29].

At the same time, in-silico design has become a common practice in research and industry. Computer simulations are routinely used in engineering design allowing for complex structures to be simulated with significant accuracy. For example, fluid flow, crack propagation and electronic simulations are routinely used in product design. Likewise in research, robust methods make it possible explore many different scenarios fairly cheaply. In spite of this, software is typically developed with particular packages in mind, and is often limited to a single domain. For example, mechanical design software might include powerful optimization procedures, but these are embedded in the packages and are optimized for particular engineering calculations such as stress analysis or crack propagation. Or, a drug docking code might contain routines that explore different drug configurations, but these are tailored to the chemistry of drug-protein interactions. As a result, users can only apply robust search methods if they are embedded in the applications of interest.

Most workflow engines, on the other hand, are designed for generalised application. They include components (which we will refer to as *actors* in this paper) that perform a wide range of activities. Such actors might form part of the dataflow of a workflow, and are used to generate, manipulate and transform data. General invocation actors can execute arbitrary software tools, making it possible to leverage existing packages. Other actors might be responsible for control flow, and route data to different actors depending on the state of the computation. However, apart from a few projects [28], workflows have not been widely used for robust design. This is partly because they are missing actors that implement design templates.

MTAGS'09, Nov 16, 2009, Portland, OR, USA.

In this paper we discuss the development of such actors, and consider a range of different design techniques – from complete enumeration to targeted search. Importantly, all of these techniques exhibit significant parallelism and can exploit highly parallel platforms by running different scenarios in parallel. We have implemented a number of new actors in an existing workflow system called Kepler [6][7][14][18][19]. When coupled with a parallel execution mechanism called TDA, we can explore multiple designs using clusters, Grids and Clouds. We demonstrate the power of a new tool called Nimrod/K to solve real problems in cardiac science.

2 ROBUST DESIGN

2.1 Techniques

Robust design involves considering more than a single data point in analysing a system. Whilst this seems like an obvious statement, it is remarkable how many scientific experiments (especially in-silico ones) only consider a single, or small number of data points. This may occur because the user may simply assume that the system is well behaved, or because any more rigorous exploration may be prohibitively expensive or infeasible.

Robust design involves understanding the parameters of a system, and exploring the design space using rigorous methods. At one end of the spectrum is complete enumeration, in which parameter ranges are broken into discrete steps, and the cross product of values is generated and explored. Complete enumeration often generates too many designs, and thus there is interest in techniques that reduce the number of scenarios evaluated in a controlled way. One of these, Fractional Factorial Design (FFD) [22], solves this by re-expressing an experimental design in terms of each of the parameters and combinations of those parameters. For example, it assumes an expression in which a set of primary terms are calculated solely on single parameter values, followed by secondary terms that concern all combinations of two parameters, tertiary terms that concern all combinations of three parameters, etc. Using this approach, it is possible to increase the resolution of the model until all combinations of all parameters are considered – this is the equivalent of complete enumeration. FFD assumes that higher order combinations have little effect on the output of a model, and thus these can be removed. The technique uses a variety of visualizations that clearly show the effects of the various parameters, and this makes it possible to prune the search space in a controlled way.

Both complete enumeration and FFD generate the scenarios statically – that is, they decompose the parameter values (and ranges) into discrete instances, and send these off for evaluation as a separate phase. When used for in-silico design, such a technique works well with commodity schedulers such as PBS, Condor, etc, where the jobs can be placed in a queue for execution. An alternative to generating all scenarios up front is to use an iterative algorithm that evaluates the quality of solutions before choosing a new set of scenarios. This mode of operation is well suited to automatic optimization, in which the goal is to minimise or maximise some objective cost function. Various optimization algorithms can be employed – ranging from heuristics that understand the design space, through to generic algorithms such as Genetic Algorithms. Optimization of this type opens up a new range of problems called inverse problems, in which the parameter values are computed to minimise the difference between some model output and a known solution. Inverse problems are common, and are also difficult to solve in general. Accordingly, it is desirable to have a range of search algorithms available.

2.2 Nimrod's approach to robust design

The Nimrod tool family automates the techniques discussed in the previous section using many-task computing (MTC) [25]. It targets computational models that are time consuming, and executes these on a range of platforms from high-end workstations to large compute clusters Nimrod consists of four related tools:

- Nimrod/G, which performs complete enumeration [5];
- Nimrod/E, which performs fractional factorial design [22];
- Nimrod/O, which performs optimization [1]; and
- Nimrod/K, which executes workflows [4].

Nimrod/G, E and O are configured by a small declarative description called a *plan* file. A plan file describes the parameters of interest, their types, values and ranges, and also a set of tasks that tell the system how to execute the model. Figure 1 shows the relationship between these three tools, and how they are exposed to users by a Web portal.

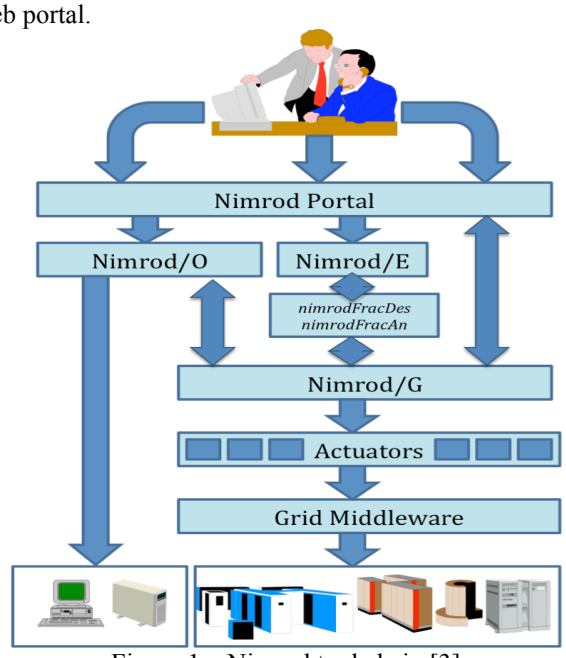

Figure 1 – Nimrod tool chain [3]

As well as functioning as a user level tool, Nimrod/G contains an API that allows other applications to schedule and execute jobs. This API, used by Nimrod/E and Nimrod/O, makes it possible to evaluate models on demand. Whilst Nimrod/E generates a set of jobs statically, Nimrod/O generates them on the fly using a range of non-linear optimization algorithms. After each batch of jobs is executed, the objective cost function values are returned to Nimrod/O, which then decides on the next set to explore. All tools exploit parallel and distributed resources by leveraging a range of Grid middleware – the most mature and commonly available interface is based on the Globus Toolkit [15][16].

Nimrod/G and Nimrod/E generate as many jobs as possible based on the resolution of the parameter specifications. Nimrod/O generates jobs in parallel batches, using parallel versions of various search algorithms where possible [1][23][24]. Nimrod/O also executes multiple searches concurrently in order to handle non-linear problems that have multiple local minima. As a consequence, all three tools generate multiple tasks. A variety of schedulers allocate jobs to resources, ranging from a first-come-first-serve heuristic that simply allocates jobs to the next available resource, through to sophisticated schedulers based on a computational economy [11].

Nimrod/K is a new version of the tool family, and is discussed in more detail in Section 3.

3 THE NIMROD/K WORKFLOW ENGINE

Nimrod/K combines scientific workflows with the robust design techniques discussed in the previous section. It builds on the Kepler workflow engine [4], which itself is built on top of the Ptolemy environment [18]. Kepler inherits a wide range of *orchestration* mechanisms from Ptolemy, enabling different programming models ranging from static dataflow to discrete event simulation. Kepler also includes an enormous range of pre-defined actors – these components can be extracted from a library and placed in workflows. Actors execute a range of functions. For example, some actors manipulate the contents of files; some invoke computations; whilst others perform data manipulation. Kepler actors are written in Java, and thus new actors can be created, filed in libraries, and invoked in new workflows. For a more complete description of the Kepler environment see [6][7][14][18][19]. Figure 2 shows a workflow as envisaged at design time, showing three actors connected by arcs. Tokens move between the actors conveying data, but the execution semantics are described by the choice of director.

Figure 2 – Static workflow of 3 actors

3.1 The Tagged Dataflow Director (TDA)

As discussed, Ptolemy directors control the way a workflow is executed. Kahn Process Networks (PN) and Synchronous Data Flow (SDF) are two of the more common directors used in scientific workflows. Both of these execute actors when data, passed as tokens, is available on actors' inputs. SDF calculates a schedule statically, whereas PN supports bounded queues of tokens and leaves actors in a ready state until tokens arrive. While powerful, neither of these directors exploit parallel platforms in a way which supports parameter sweeps and search. For example, SDF only fires one actor at a time, regardless of how many tokens are active in a workflow. Thus, even when multiple tokens are queued, the workflow cannot exploit this concurrency. PN, which does allow some inter-actor concurrency, only runs one copy of each actor at a time. When used to perform robust design, however, we want to generate a large number of scenarios and execute these in parallel. Accordingly, we want the number of instances of any given actor to increase dynamically as more data tokens are sent to it.

To solve this problem, we created a new director called TDA. TDA implements a Tagged-Dataflow Architecture originally designed for the MIT, RMIT and Manchester data flow machines [17][2][9]. TDA augments data tokens with a tag, or colour field. When multiple tokens of different colours are available on the inputs of an actor, multiple independent instances of the actor are invoked. Thus, by controlling the colouring and de-colouring of token, the workflow can scale to use an arbitrary number of processors. Figure 3 shows the flow of tokens in this workflow when only one token is present on the arcs – 3(a) shows a token on the input of actor 2, and 3(b) shows the token emitted by actor 2 after it has executed. Figure 4 shows the flow of multiple coloured tokens on arcs – 4(a) shows 3 tokens on the input to actor 2, and 4(b) shows the multiple copies of that actor running in parallel.

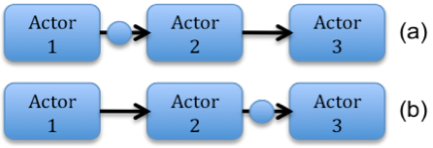

Figure 3 – Tokens in static workflow

The TDA implementation leverages a feature of Ptolemy that "clones" actors. The TDA maintained a set of queues for each actor. When multiple tokens of different colour arrive on an actor's input, TDA creates new copies of the actor using cloning. These clones are destroyed after execution, which allows the graph to dynamically expand and contract as the concurrency changes. More details are discussed in [4].

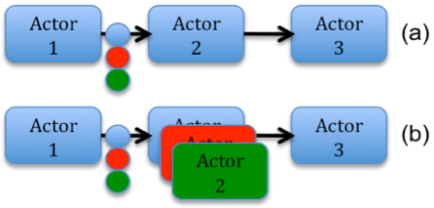

Figure 4 – Tokens in dynamic workflow

3.2 Nimrod/K Actors

Three new families of actors have been created to support the robust design methods discussed in Section 2.

3.2.1 Nimrod/G Actors

As discussed, Nimrod/G takes a cross product parameter values and computes a set of scenarios. Parameter ranges and values are normally provided in a plan file, although they can be added dynamically through the Nimrod/G API. In Nimrod/K, we have created a new actor that takes the same parameter descriptions used in plan files, and it emits a series of tokens – one per instance.

Figure 5 shows the way this actor is instantiated in a workflow, and also shows the configuration of the parameter values. In this case, ParameterSweep is linked to a MatlabExpression actor that executes a Matlab program. The tokens that flow into the Matlab actor contain actual values for the various parameters – in this case 9 different variables. This actor works with the current Ptolemy directors, however, when used in combination with the TDA, parallel execution of the different parameter combinations allows the workflow to execute efficiently on the Grid.

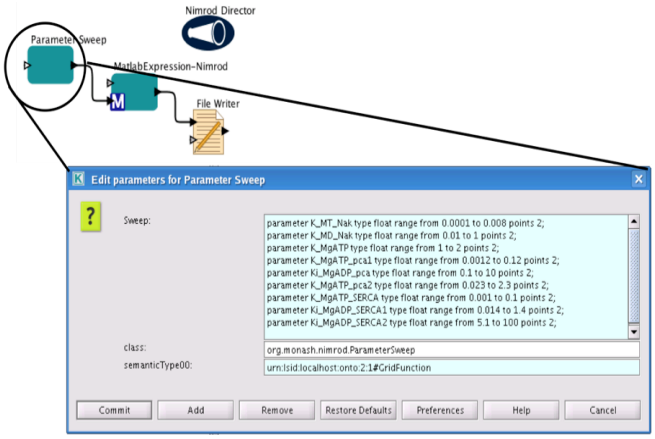

Figure 5 – Nimrod/G actors and configuration control

3.2.2 Nimrod/E Actors

Nimrod/E is similar to Nimrod/G in that it generates a set of experiments. However, it uses the Fractional Factorial Design algorithm to decide which combinations to explore. These are produced by the ExperimentalDesign actor. The plan file syntax for Nimrod/E is a superset of Nimrod/G, and contains additional statements that control the resolution of the model. In addition to the ExperimentalDesign actor, an EffectsEstimator actor computes the effects of the various parameter combinations. These are streamed to two different graphing actors, one that generates a Daniel plot, and another that generates a Lenth plot. These two diagrams show the effects of various parameter combinations in a graphical form. Lenth plots show the effects of various parameter combinations. Parameter combinations, which are shown as individual dots, are plotted using the Y axis range to indicate their contribution. Thus, dots that lie close to the X axis have little effect on the output. In the Daniel plots, the results are displayed using a cummulative graphing technique. Here, insignificant combinations cluster on a straight line, and significant ones appear off the line. Using these two forms of display, it is very easy to spot the significant combinations. Figure 6 shows a workflow that uses all these actors. Figure 7 shows examples of Daniel and Lenth plots.

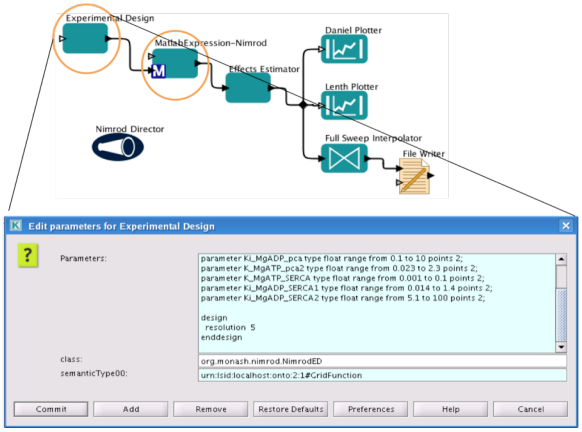

Figure 6 – Experimental Design actors

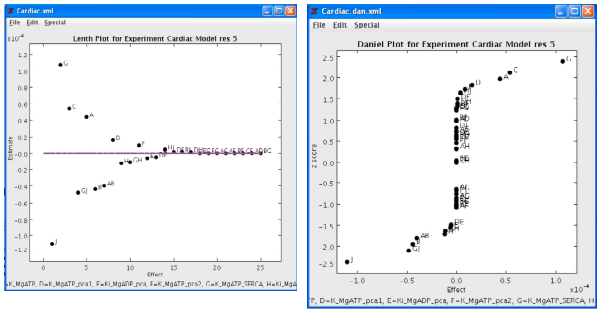

Figure 7 (a) Lenth Plot (b) Daniel Plot

3.2.3 Nimrod/O Actors

Optimization algorithms combine well with workflows because they usually involving repetitive looping in which results are passed from one iteration to the next. Implementing Nimrod/O functionality within Kepler

exposes the components of the optimization process, giving the user a clear view of the data flows and facilitating substitution of these components. Accordingly, we have built a set of actors that support the generic actors of an optimization algorithm, and also specific actors that implement specific algorithms.

Figure 8 shows a generic optimization algorithm and the actors that support it. First, the domain of the search is defined by specification of the input parameters using a similar method to Nimrod/G plan file syntax. Within this domain the next component will select starting points for the multiple optimizations. Each starting point will begin an optimization run using the specific optimizer actor. This actor generates sets of points that are sent for evaluation, a "batch of jobs" in Nimrod/O's terminology. Running the models represented by these jobs produces numerical results that are used to decide the next generation of points. This cycle continues until some convergence criterion has been achieved whereupon a final point is forwarded as the result of the search. Figure 8 also shows a component that evaluates constraint conditions imposed by the user. This calculates the margin by which a search point violates any constraints, and imposes penalties accordingly. If the constraint is a "hard" one then the point is completely forbidden, obviating the need for model evaluation.

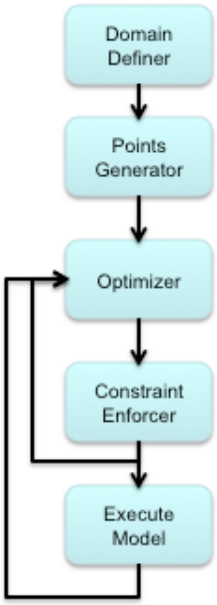

Figure 8 – Generic optimization actors

We are currently building actors for the existing Nimrod/O optimization methods, namely gradient descent, simplex, genetic algorithms and simulated annealing.

4 CASE STUDIES: CARDIAC IONINC-METABOLIC MODEL

In this section we illustrate the three design techniques in Nimrod/K using a real world case study in cardiac modelling. The electrical activity of the heart is based on cycles of ion transfer via cell surface membrane. There has been much research on developing robust and accurate models of myocyte behaviour so that in-silico experiments can be performed on complete cardiac systems. The research has significance from basic science through to therapeutic drug design.

Cardiac excitation-contraction coupling is a sequence of well-orchestrated events. Both excitation and contraction processes and their interactions are required for an integrative model of cardiac electromechanical interactions [26]. A number of mathematical models have been developed to study excitation-contraction coupling in ventricular cardiac cells. Recently, two of the authors of this paper integrated additional ionic-metabolic equations [8][20] into the Shannon et al. four compartment cell model [26]. The updated ionic model is more complex due to the multiple domains and increased number of unknown model parameters, so it implies a less stable set of ordinary differential equations systems. For this reason, any information gained based on this model can yield a significant level of misunderstanding if one wants to use it investigate the excitation-contraction coupling in ventricular myocytes.

As a result of this integration, we are interested in two different experiments. First, we want to calibrate the new model so that it matches real physiological data. Second, we want to explore the sensitivity of the model to the parameters, and determine whether there are some combinations that are more important than others. Accordingly, the first experiment is performed with Nimrod/OK and the second with Nimrod/EK.

4.1 Nimrod/OK Experiment

After adding the new channel and metabolic factors for both normal condition and ischemia, we confirm the results against physiological data. We then stabilize the model for a heart rate of 0.5Hz and a longer performance duration (3 min). In both, the focus is on calcium transient concentrations (dyad, sub-membrane space, cytosol, sarcoplasmic reticulum), and action potential shape and duration, to determine the stability and accuracy of our model. In order to validate the model, some final outputs are compared against experimentally known data [20][26]. This involved exploring a range of numbers for nine input metabolic constants and selecting the combination that produces the closest output to the desired value.

This experiment performed simplex searches over the nine-dimensional space, using the workflow shown by Figure 9. The computations are performed on a distributed Grid of Linux clusters, as shown in Table 1.

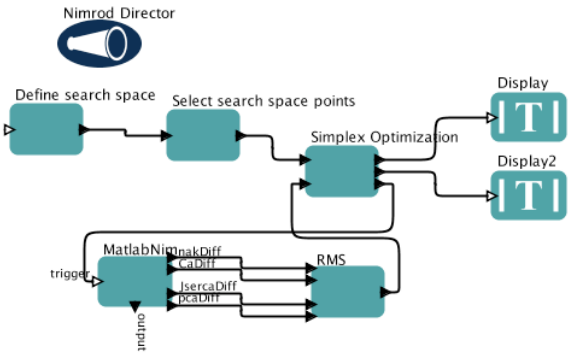

Figure 9 – Optimization run in Nimrod/OK.

Machine	# Cores	Hardware	Location
East	160	Intel(R) Xeon(R) E5310 @ 1.60GHz	Monash University, Melbourne
	64	Intel(R) Xeon(R) 5160 @ 3.00GHz	
West	160	Intel(R) Xeon(R) E5310 @ 1.60GHz	Deakin University, Geelong
South	160	Intel(R) Xeon(R) E5310 @ 1.60GHz	RMIT, Melbourne

Table 1 – Properties of Grid Testbed

128 starting points are randomly selected within the domain. Table 2 summarises the results of these searches, showing the best, worst and average search results. We also show the number of jobs and batches required to achieve the best and worst results, and the average number of jobs and batches. The best search gives a considerably better (smaller) result than the average, at the expense of more computational effort as shown by the numbers of jobs and batches of jobs. This shows the utility of multiple searches.

Job executions averaged 3m 21s, but with substantial parallelism the full experiment was completed in 7h 37m. Figure 10 plots the number of jobs executing over the duration of the experiment. Initially the simplex method evaluates the objective at the vertices of the starting simplex. In this case each of the simplexes has ten such vertices, so the experiment began with 1280 jobs. Thereafter, most iterations require just four new evaluations, so the load dropped to 512 jobs, except for the occasional peak where a new simplex was required. As searches completed the load dropped in stages; the final longest running optimization required only four processors but dominated the experiment wall clock time.

Index	Objective	Jobs	Batches
Min	0.00042	154	37
Average	0.0015	142	35
Worst	0.0038	34	7

Table 2 – results for searches over the full domain

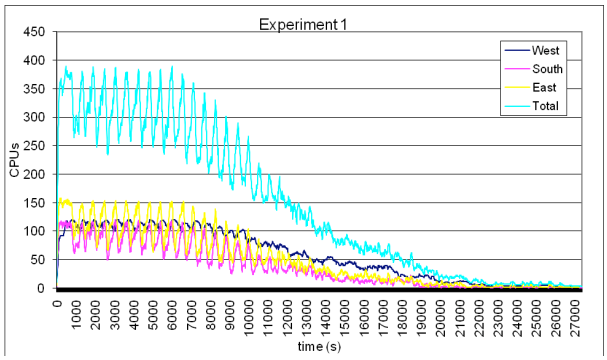

Figure 10 – job concurrency for full domain search

4.2 Nimrod/EK Experiment

The main goal of this experiment was to examine the changes in which total ionic flux(s) in the compartment of interest may play the largest role in determining specific model outputs (action potential (AP) and Ca transients in the four cell sub-domains) in ventricular cardiac cells isolated from the sub-endocardium tissue layer in rabbits. We use Nimrod/EK to perform a parameter sweep and measure the effects of each of the parameter combinations, as summarised by the workflow in Figure 11.

In agreement with experiment [26], results suggest that small changes in L-type calcium and sodium/calcium exchanger total fluxes in the junctional cleft may significantly affect the cell function. Specifically, Figure 12 shows a Lenth plot for the AP duration. Here, there are few combinations that actually exhibit any significant effect because most of them are clustered around the X axis. This is reenforced in the Daniel plot in Figure 13, in which significant combinations appear off the line, and those on the line are insignificant. The same effects are apparent in the cytosolic Ca-peak in Figures 14 and 15. Surprisingly, the studies suggested that small changes in the junctional chloride/calcium flux also may have prominent effect on the model outputs. New experiments need to be perfomed to test this hypothesis. The fact that both the action potential duration and cytosolic calcium peak are impacted by the same fluxes distributions strengthen further our findings.

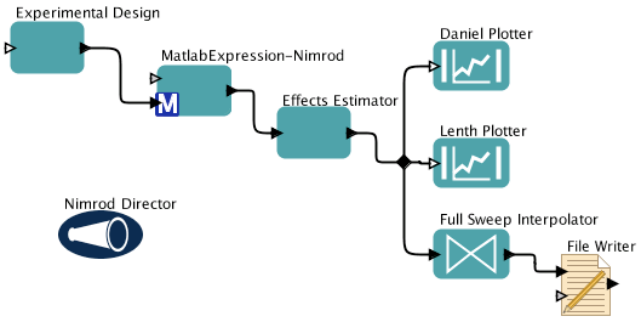

Figure 11 – workflow using Nimrod/EK

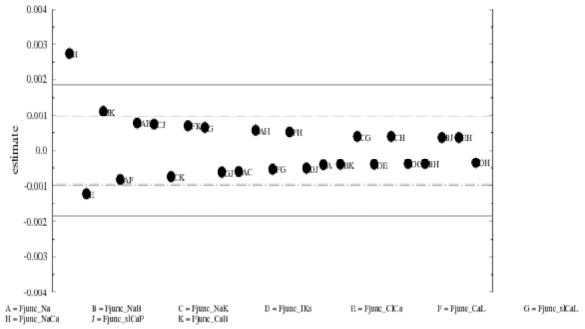

Fig. 12 Length plot for AP duration

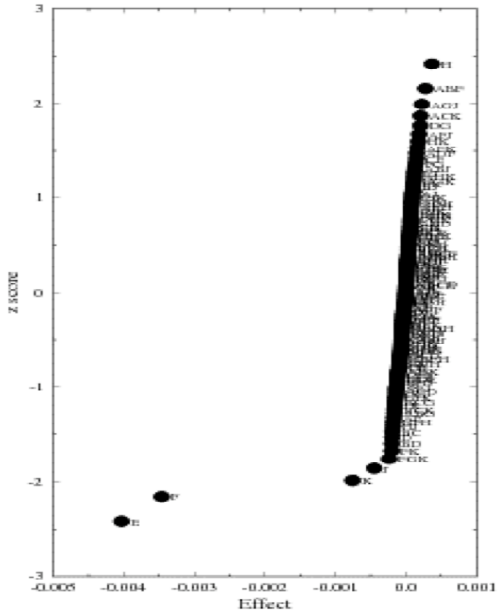

Fig. 13 Daniel plot for AP duration comparison

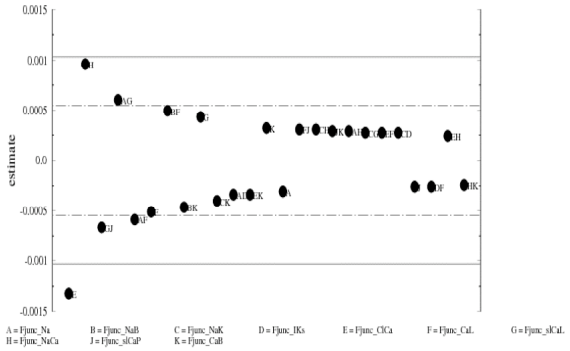

Fig. 14 Length plot for cytosolic Ca-peak comparison

5 RELATED WORK

While most workflow engines support some form of Multi-task Computing, many of them are remarkably static – that is, the graphs that are generated do not typically change size at run time and thus the level of parallelism cannot be varied easily. This static code generation strategy means there are usually only two ways to specify parallel activity. Either,

1. build a very large static DAG with cloned sub-graphs for each parallel activity; or
2. write a workflow that contains logic to spawn parallel activities.

The first of these means that the amount of parallelism cannot change once the graph has started execution, which is restrictive. Further, if there is a lot of parallelism, the graph can become very large and difficult to manage. For example, a workflow that wishes to process the contents of a database, in parallel, must be replicated many times to allow them to run in parallel. The underlying graph can become very large, but it also contains a high degree of redundancy since the same operations are applied to each database element. This is the approach taken with tools like APST [12] and Pegasus/DAGMAN [13]. The second alternative supports dynamic parallelism, but pushes the onus onto the programmer to add the task management logic. For example, users must explicitly code for parallel execution, and incorporate loops that process the contents of the database, provide mechanisms that spawn independent calculations and then synchronize the results. This process significantly complicates the task of writing a workflow. In spite of this, the approach is typical of systems such as Kepler, for example.

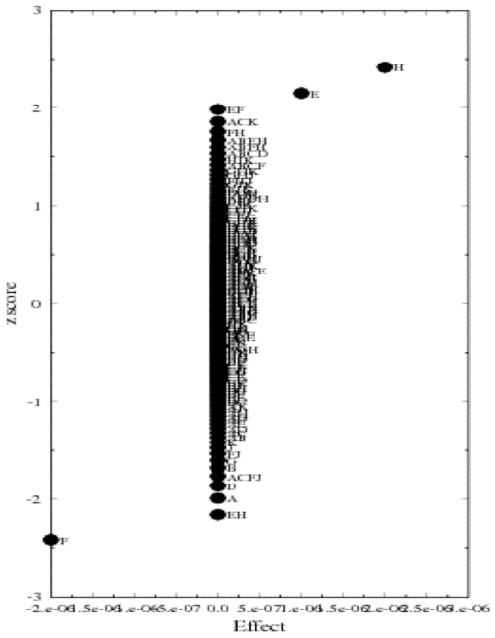

Fig. 15 Daniel plot for cytosolic Ca-peak comparison

An alternative to static code generation is to support the dynamic spawning of sub-graphs. This is the approach effectively taken in Nimrod/K, Swift [30] and the most recent additions to Taverna [21]. These systems are simpler to use, because the task management code is embedded in a

middleware layer, and they can scale from small to large amounts of parallelism very quickly and dynamically.

Almost no workflow engines support the search optimization algorithms implemented in Nimrod/O, with the notable exception of Geodise [28]. Geodise's goal is to "expose optimisation services in a flexible, generic interface that can be easily integrated into various environments and used to compose different optimisation workflows." [28]. In Geodise, optimization algorithms are exposed as Web services, which can be invoked by a workflow system, or any other programming language that can execute Web service calls. In our case we have embedded our work in an existing workflow engine, Kepler, and have built Kepler actors that perform the optimization functions.

6 CONCLUSIONS

This paper has provided a historical view of the Nimrod tool family, and more importantly, showed how we have leveraged an existing workflow engine to incorporate Nimrod's advanced search techniques. Specifically, we have added complete enumeration, fractional factorial design and optimization methods to Kepler by building special actors. In order to support parallel execution, we have incorporated a new Kepler director called TDA. The result is a very flexible workflow environment that can be used for automatic design. Importantly, we leverage the enormous number of existing actors in Kepler and Ptolemy.

We have demonstrated the applicability of Nimrod/K by various case studies in cardiac science. These have shown that we can solve a complex inverse problem framed as an optimization problem, and then using the Nimrod/OK actors. We have also used Nimrod/EK to explore the role of various parameters in the mathematical model.

Nimrod/K is an active research project. We are currently exploring a number of research themes using this as a base. First, we are building a data transport framework that removes the need for users to explicitly manage data staging and movement in their workflows. When combined with the Nimrod/K engine, this will provide a very powerful distributed computing platform. Second, we are designing and building arrange of scheduling heuristics that work with the data transport and parallel computing models discussed [10][27]. Third, we are building interfaces with powerful display technologies, and these allow a user to be immersed in the optimization process.

ACKNOWLEDGMENTS

This project is supported by the Australian Research Council under the Discovery grant scheme We acknowledge our colleagues in the MeSsAGE Lab at Monash University and the Cardiac Mechanics Lab at UCSD. We also acknowledge the US NSF funded PRIME project, (supporting Amirriazi and Chitters) and thank colleagues, in particular Peter Arzberger, for their strong support.

REFERENCES

[1] Abramson D, Lewis A, Peachey T, Fletcher, C., "An Automatic Design Optimization Tool and its Application to Computational Fluid Dynamics", SuperComputing 2001, Denver, Nov 2001

[2] Abramson, D and Egan, G, "The RMIT Dataflow Computer: A Hybrid Architecture", The Computer Journal, Vol 33, No 3, June 1990, pp 230 - 240.

[3] Abramson, D., Bethwaite, B., Enticott, C., Garic, Slavisa and Peachey, T. "Parameter Space Exploration using Scientific workflows", ICCS 2009, SU Center for Computation & Technology, Baton Rouge, Louisiana, U.S.A, May 25-27, 2009.

[4] Abramson, D., Enticott, C and Altinas, I. "Nimrod/K: Towards Massively Parallel Dynamic Grid Workflows", IEEE Supercomputing 2008, Austin, Texas, November 2008.

[5] Abramson, D., Giddy J., and Kotler, L. "High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid," In *Int'l. Parallel and Distributed Processing Symposium (IPDPS),* Cancun, Mexico, May 2000. http://www.csse.monash.edu.au/~davida/nimrod/.

[6] Altintas, I. Berkley, C. Jaeger, E. Jones, M. Ludäscher B. and Mock, S. "Kepler: Towards a Grid-Enabled System for Scientific Workflows," in the *Workflow in Grid Systems Workshop in GGF10* - The 10th Global Grid Forum, Berlin, March 2004.

[7] Altintas, I., Birnbaum, A., Baldridge, K., Sudholt, W., Miller, M., Amoreira, C., Potier Y. and Ludaescher, B. "A Framework for the Design and Reuse of Grid Workflows" *Intl. Workshop on Scientific Applications on Grid Computing* (SAG'04), LNCS 3458, Springer, 2005

[8] Amirriazi S., Chang S., Peachey T., Abramson D. and Michailova A., *Optimizing Cardiac Excitation-Metabolic Model By Using Parallel Grid Computing*, Biophysics 2008, Long Beach, California, February 2008

[9] Arvind, and Nikhil R.S. "Executing a Program on the MIT Tagged-Token Dataflow Architecture", IEEE Transactions on Computers, Vol. 39, No. 3 (1990)

[10] Ayyub, S. and Abramson, D. "GridRod - A Service Oriented Dynamic Runtime Scheduler for Grid Workflows". 21st ACM International Conference on Supercomputing, June 16-20, 2007, Seattle.

[11] Buyya, R. and Abramson, D. "The Nimrod/G Grid Resource Broker for Economic-based Scheduling", in Market-Oriented Grid and Utility Computing (Wiley Series on Parallel and Distributed Computing), Editors Rajkumar Buyya and Kris Bubendorfer, ISBN-13: 978-0470287682, November 2009.

[12] Casanova H. and Berman, F. "Parameter Sweeps on The Grid With APST", chapter 26. Wiley Publisher, Inc., 2002. F. Berman, G. Fox, and T. Hey, editors.

[13] Deelman E., Blythe J., Gil Y., Kesselman C., Mehta G., Patil S., Su M, Vahi K., Livny M., "Pegasus : Mapping Scientific Workflows onto the Grid", Across Grids Conference 2004, Nicosia, Cyprus.

[14] Eker, J., Janneck, J. W., Lee, E. A., Liu, J., Liu, X., Ludvig, J., Neuendorffer, S., Sachs, S., Xiong, Y., "Taming Heterogeneity---the Ptolemy Approach," Proceedings of the IEEE, v.91, No. 2, January 2003.

[15] Foster I. and Kesselman, C. “Globus: A Metacomputing Infrastructure Toolkit,” *Int’l J. of Supercomputer Applications*, vol. 11, no. 2, 1997, pp. 115-128.
[16] Foster, I., “Globus Toolkit Version 4: Software for Service-Oriented Systems,” Conference on Network and Parallel Computing, 2005.
[17] Gurd J.R. and Watson I. “Data Driven System for High Speed Parallel Computing (1 & 2) Computer Design, vol.9 nos.6 & 7, June & July 1980, pp. 91-100 & 97-106.
[18] Lee, E. et al, "Overview of the Ptolemy Project," Technical Memorandum UCB/ERL M01/11, University of California, Berkeley, March 6, 2001.
[19] Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger-Frank, E., Jones, M., Lee, E., Tao J. and Zhao, Y. “Scientific Workflow Management and the Kepler System”, *Concurrency and Computation: Practice & Experience*, Special Issue on Scientific Workflows, 2005.
[20] Michailova A., Lorentz W., McCulloch A., *Modelling Transmural Heterogeneity of KATP current in Rabbit Ventricular Myocytes*, AJP-Cell Physiology, 293 (2007), pp C542-C557.
[21] Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M. R., Wipat, A., Li, P., Taverna: A tool for the composition and enactment of bioinformatics workflows. Bioinformatics Journal, 20(17):3045–3054, 2004.
[22] Peachey, T. C., Diamond, N. T., Abramson, D. A. Sudholt, W., Michailova, A. Amirriazi, S., *Fractional Factorial Design for Parameter Sweep Experiments using Nimrod/E,* Journal of Scientific Programming, Volume 16, Numbers 2,3, 2008.
[23] Peachey, T., Abramson, D. and Lewis, A. *Model Optimization and Parameter Estimation with Nimrod/O,* The International Conference on Computational Science, May 28-31, 2006, University of Reading.
[24] Peachey, T., Abramson, D., Lewis, A., Kurniawan, D. and Jones, R. *Optimization using Nimrod/O and its Application to Robust Mechanical Design,* PPAM 2003, Czestochowa, Poland, Lecture Notes in Computer Science, Volume 3019 / 2004, pp. 730 – 737, September 7-10, 2003.
[25] Raicu, I., Foster, I., Zhao. Y. "Many-Task Computing for Grids and Supercomputers", IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS08), 2008.
[26] Shannon T. R., Wang, F. Puglisi, J. Weber C. and Bers, D. M. *A Mathematical Treatment of Integrated Ca Dynamics within the Ventricular Myocyte*, Biophysical Journal, 87 (2004), pp 3351-3371.
[27] Smanchat, S., Indrawan, M., Ling, S., Enticott, C. and Abramson, D. “Scheduling Multiple Parameter Sweep Workflow Instances on the Grid”, IEEE e-Science 2009, Dec 9 – 11th, Oxford, UK.
[28] Xue G., Song W., Cox S.J., and Keane A.J., Numerical Optimisation as Grid Services for Engineering Design, Journal of Grid Computing, Vol.2 No.3, 2004, pp223-238.
[29] Yu J., and Buyya, R., A Taxonomy of Workflow Management Systems for Grid Computing”, Journal of Grid Computing, Springer Press, New York, USA.
[30] Zhao, Y., Hategan, M., Clifford, B., Foster, I., von Laszewski, G., Raicu, I., Stef-Praun, T. and Wilde, M. “Swift: Fast, Reliable, Loosely Coupled Parallel Computation”, IEEE Workshop on Scientific Workflows 2007.