Beatbox—A Computer Simulation Environment for Computational Biology of the Heart

Ross M^{c}Farlane, Irina V. Biktasheva
University of Liverpool
Computer Science Department
Ashton Building
Ashton Street
Liverpool
L69 3BX, UK
{*rossmcf, ivb*}*@liv.ac.uk*

Abstract

Despite over a century's study, the trigger mechanisms of cardiac arrhythmias are poorly understood. Even modern experimental methods do not provide sufficient temporal and spacial resolution to trace the development of fibrillation in samples of cardiac tissue, not to mention the heart *in vivo*. Advances in human genetics provide information on the impact of certain genes on cellular activity, but do not explain the resultant mechanisms by which fibrillation arises. Thus, for some genetic cardiac diseases, the first presenting symptom is death.

Computer simulations of electrical activity in cardiac tissue offer increasingly detailed insight into these phenomena, providing a view of cellular-level activity on the scale of a whole tissue wall. Already, advances in this field have led to developments in our understanding of *heart fibrillation* and *sudden cardiac death* and their impact is expected to increase significantly as we approach the ultimate goal of whole-heart modelling.

Modelling the propagation of *Action Potential* through cardiac tissue is computationally expensive due to the huge number of equations per cell and the vast spacial and temporal scales required. The complexity of the problem encompasses the description of ionic currents underlying excitation of a single cell through the inhomogeneity of the tissue to the complex geometry of the whole heart. The timely running of computational models of cardiac tissue is increasingly dependant on the effective use of *High Performance Computing* (HPC), i.e. systems with parallel processors. Current state of the art cardiac simulation tools are limited either by the availability of modern, detailed models, or by their hardware portability or ease of use. The miscellany of current model implementations leads many researchers to develop their own ad-hoc software, preventing them from both utilising the power of HPC effectively, and from collaborating fluidly. It is, arguably, impeding scientific progress.

This paper presents a roadmap for the development of *Beatbox*, a computer simulation environment for computational biology of the heart—an adaptable and extensible framework with which High Performance Computing may be harnessed by researchers.

Keywords: Computational biology, Cardiac electrophysiology, High Performance Computing

1. BACKGROUND

Cardiovascular disease (CVD) is the main cause of death in Europe, accounting for 48% of all deaths (British Heart Foundation Health Promotion Research Group 2008). Cardiac arrhythmias, in which the electrical activity of the heart responsible for its pumping action is disturbed, form some of the most serious. Despite over a century's study, the circumstances from which such fatal cardiac arrhythmias arise are still poorly understood. Although several advancements have been

made in linking genetic mutations to arrhythmogenic CVD (Clancy and Rudy 1999; Noble 2002*b*; Veldkamp, Viswanathan, and Bezzina 2000), the mechanisms by which arrhythmias emerge remain elusive.

The position of the heart in the torso makes *in vivo* measurement awkward and invasive, prohibitively so for study in humans. *In silico* simulations can allow unimpeded access to the whole heart, including intramural tissue, with greater spacial and temporal resolutions than wet experiments. Also, when studying diseases for which the first presenting symptom is often death, there is an understandably limited opportunity to make even superficial examinations *in vivo*. Computer simulations allow us to synthesise such elusive phenomena for closer study.

Useful simulations of cardiac tissue involve vast scale ranges, with activity at μm and μs scales being observed in several cm^3 of tissue over tens of seconds. Moreover, the rising popularity of biophysically realistic cell models, which calculate quantitative data, increases the volume of data to be computed and stored for each point in space and time. The complexity of simulations are further increased as simulations expand 'outward' to include greater detail at the cellular level, such as cell metabolism, and greater integration with surrounding biological systems, such as vascular fluid dynamics. In this context, it is unsurprising that the timely completion of simulations relies on modern high performance computing (HPC) hardware.

Use of HPC facilities, although essential, is limited by the complexities of software development; a mixture of hardware architectures and associated programming models hamper development. This can be compounded in situations where funding is more easily obtained for hardware purchase than for the accompanying skilled staff. The result is that many small software development projects are undertaken in isolation, each supporting a particular short-to-medium-term aim. Code developed in this environment is often poorly structured and documented—hampering reuse within the lab—and is unlikely to be portable, preventing use on different platforms elsewhere. Moreover, results produced from such software are difficult to compare or repeat, limiting the scope of peer review. This rather insular approach to development excludes a great many members of the community for whom code-level involvement with simulations is neither desirable nor practical. The disparity of cardiac simulation software at present is, arguably, impeding scientific progress.

The project described in this paper is developing **Beatbox**, an alienable computer simulation environment. Beatbox expands on the core code of the QUI software package, developed at the Russian Academy of Sciences (Biktashev and Karpov Unpublished). Foremost, Beatbox will be able to harness HPC on a broad range of computing platforms, abstracting much of the complexity associated with HPC development away from users and model developers. Users with little programming knowledge will have the ability to simply configure and run simulations, while those more proficient with software development will be able to extend Beatbox via robust and flexible interfaces. It is hoped that, with the help of the community, Beatbox will encourage greater intra- and interdisciplinary collaboration and co-operation.

2. COMPUTING THE ELECTRICAL ACTIVITY IN THE HEART

To simulate the heart is to describe and predict its electro-mechanical activity over time. The rhythmic contraction of the heart is coordinated by a wave of electrical excitation, called *Action Potential* (AP), which propagates through the muscle tissue from autorhythmic cells in the heart's in-built pacemaker, the *sino-atrial node*. The sequence of contractions that form a single heart beat relies upon the smooth propagation of AP through excitable tissue. Arrhythmias are disturbances to this propagation. Of particular interest to researchers are *reentrant arrhythmias*, in which AP propagation enters a re-entrant circuit (aka spiral wave). In *fibrillation*, turbulence in re-entrant waves of excitation causes the muscle to flutter chaotically rather than contract rhythmically. Patients with fibrillation in the ventricles are unlikely to live for more than a few minutes without intervention.

Most cardiac models are constructed “middle out” (Noble 2002*a*), working from an intermediate level of detail, the cell, gradually integrating the smaller scale physiological events on which the model is based and the macroscopic processes with which cells interact. The majority of models are formulated as *Reaction-Diffusion* equations, where the reaction component describes local cell kinetics and the diffusion component describes their interconnection.

The description of a single *cardiomyocyte* (heart muscle cell) forms the basis of most models and is the focus of much research activity. While early, *phenomenological* models, e.g. (FitzHugh 1961) were capable of mimicking the behaviour of cells qualitatively to a reasonable degree of accuracy, there is an increasing demand for models to provide quantitative values of chemical concentrations in the cell, based on the underlying electrophysiology.

Modern models, e.g. (Courtemanche, Ramirez, and Nattel 1998)(Luo and Rudy 1994) improve upon early automata-based models (Wiener and Rosenblueth 1946) and are able to produce useful quantitative data, by representing the states of living cells with continuous values, capable of changing at varying rates over continuous time. Mathematically, a cell model consists of a system of *ordinary differential equations* (ODEs), describing the time course of the membrane potential and ionic currents through the membrane. Modern models of cardiac cells comprise in excess of 20 ODEs per cell. For computation, ODEs are discretised using time-steps sufficiently fine-grained to capture the rapid changes that take place in the early phases of action potential. Ordinarily, time-steps in the order of tens of μs will be used to describe phenomena emergent over tens of seconds.

Most tissue models use a *monodomain* description, where the ‘bath’ of ions outside the cell is assumed to be static, with no changes in chemical concentration or potential. Alternatively, bidomain models (Henriquez and Papazoglou 1996) describe the intra- and extra-cellular spaces discretely. Bidomain models are increasing in popularity due to their improved physiological realism.

The interconnectivity of cells is characterised by the diffusion component of the reaction-diffusion equation. Unlike an electrical cable, conduction through muscle tissue is not the passage of electrons, but the propagation of state: like a Mexican wave on a sporting grandstand, no energy is exchanged between participants, instead each person supplies their own energy following a stimulus from their neighbour. The diffusion component defines the conductivity of the medium, that is the rate at which membrane potential diffuses between cells which, together with the kinetics of individual cells, defines the speed of AP propagation.

The diffusion component also defines the neighbourhood of a cell; the surrounding cells from which stimulus may come. In relatively simple simulations, a cell will have equal diffusion to and from each of its neighbours. *In vivo*, however, tissue is *anisotropic*, meaning that its diffusion will vary with direction. Quite how the diffusion varies is dependent on the position of the cell and each of its neighbours in the structural geometry of the muscle. In cardiac muscle, cells are arranged into fibres, with conduction along the fibre axis much faster than in the transverse direction. As shown in Figure 1, fibres lie in parallel to form sheets. Sheets are then layered, each with a slight rotation, to form a slab of muscle.

Simulation of the heart’s anisotropy requires a model of its geometry in order to relate each cell to fibre directions and sheet planes. Data for such a model are provided by detailed measurement of hearts, either by isolating and slicing a mammalian heart (LeGrice 2001), or by Diffusion Tensor Magnetic Resonance Imaging *in vivo* (Hsu et al. 1998). Once captured, the model must then be discretised for computation using either a finite differences (FD) or finite elements (FE) technique. In FD, the medium is discretised using a Cartesian mesh. Implementation of FD systems is straightforward, since neighbourhoods are easily defined and even anisotropic diffusion may be described with relative ease. FE represent an ‘engineering’ approach to cardiac modelling. FE techniques divide the space into triangles (2D) or tetrahedra (3D) of varying size, computed to fit the curvature of the medium.

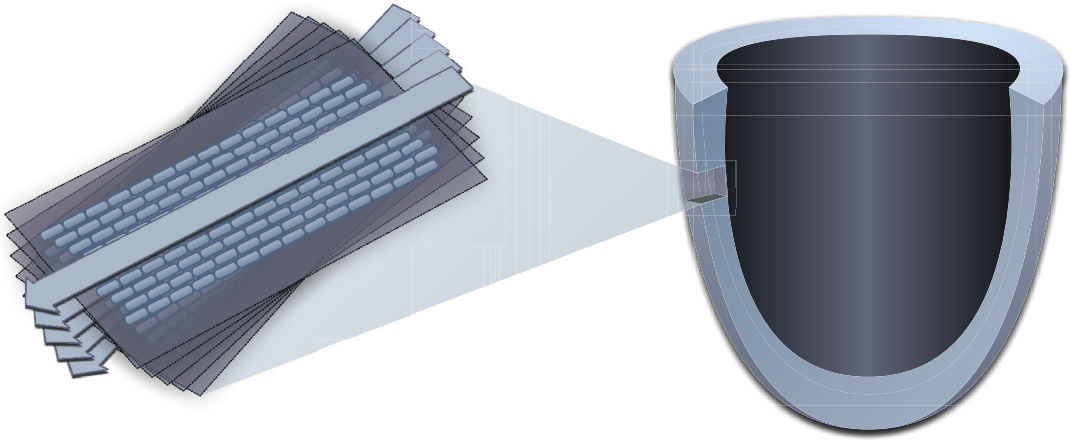


FIGURE 1: Cardiac Tissue Geometry

The equations and simulation components described above represent the 'middle' of simulation scope at present. As our knowledge deepens and techniques and tools improve, the horizons of simulations are broadened to include additional micro- and macroscopic processes, some of which are described below:

Extending scope 'downwards', some models increase the detail with which they describe single cells by including lower-level functions, such as metabolism. This kind of detail allows us to recreate the ischemia caused as arrhythmias reduce the heart's ability to circulate oxygenated blood, which in turn affects cardiac muscle's ability to contract.

Looking 'upwards', Reaction-Diffusion systems may be expanded to include muscle mechanics, where cell state is related to myocytes' physical contraction. In the simplest models, contraction may be linked directly to transmembrane potential, whereas more complex physiological models compute contraction using intracellular calcium dynamics, as *in vivo*. Taken further, models may include *mechano-electric feedback*, in which tissue deformations affect AP via stretch-activated channels (Panfilov, Keldermann, and Nash 2007).

Broadening the horizons of simulations further, some work has already been done to integrate a whole-heart model into large-scale models of surrounding systems. For example, (Sundnes et al. 2006) describes inclusion of a whole-heart model into a coarse model of the torso, enabling the creation of a virtual ECG. Such a simulation allows researchers to relate electrophysiological phenomena to their presentation in a clinical context.

3. HIGH PERFORMANCE COMPUTING

As we have seen, simulating the heart requires complex, non-linear equations to be solved across vast spacio-temporal scales. Even simple simulations can require weeks of run time. The timely execution of simulations relies upon effective use of High Performance Computing (HPC) facilities. Simulations may be accelerated by sharing the computational load amongst processors, usually by *domain decomposition*, assigning an area of the simulation medium to each processor. Many of the challenges of developing software for HPC surround the interaction of each thread or process, constrained by their access to memory. Programming models, design considerations and terminology differ with the architecture of the cluster's hardware.

Clusters can be grouped into two broad categories, based on the arrangement of their memory. In a *shared memory* cluster, processors have uniform access to memory, allowing each processor direct access to the application data, but requiring careful management, so as to maintain

data integrity. The physical limitations and cost of such systems effectively limit the number of processors available to them and thus their popularity.

Due to their affordability and scalability, *distributed memory* clusters are far more common. In most cases, these are formed of several standard server computers connected using a high speed network. As the name suggests, memory is distributed across machines in the network, with each processor able to access only its own, local memory. Co-ordination requires explicit communication between processes, which must be minimised to optimise performance.

The proliferation of multicore processor chips as commodity components means that many modern machines are shared memory systems in their own right. When networked to form a distributed memory cluster, this presents us with a new, hybrid, architecture. Desktop users benefit from multicore technology when running multiple applications. Scientific computing, on the other hand, relies on one application doing the bulk of the work, and must therefore utilise this added parallelism explicitly. Optimising software for hybrid architectures presents new challenges for developers, in balancing the two-tiered parallelism it provides. Moreover, a general-purpose simulation tool will need to strike this balance without *a priori* knowledge of hardware performance or the simulation task.

4. COMPUTER SIMULATION ENVIRONMENT

Despite the interest it may generate for computer scientists, computer simulations are a means to an end for computational biologists, whose time would almost certainly be better spent devising simulations or analysing their results than struggling with their implementation. Coding a complete simulation represents an unnecessary reinvention of the wheel, while delving into existing code presents a steep learning curve, even to technically proficient users. Add to this the additional complexity of software development for HPC, and it becomes easy to understand the impediment created by what should be supporting technologies.

At present, much of the software development in the field is written in an ad-hoc fashion, by and for a single lab to serve or describe a particular experiment. The software will usually be written with just one platform in mind, without any consideration for how it might run elsewhere. With code maintained by its authors, documentation is often overlooked, impeding the involvement of new users and developers. Adding new functionality to code of this kind usually requires an in-depth knowledge of its design and structure, which is again difficult to ascertain without sufficient documentation.

For many pieces of internally developed simulation software, the only means of user interaction is the alteration of hard-coded simulation parameters. The cycle of edit, recompile, debug, repeat, is not only tedious, but exposes low-level program code to front-end users. It should come as little surprise that code developed in this manner will make at best trivial use of HPC.

5. THE BEATBOX PROJECT

The Beatbox Project at The University of Liverpool is currently working towards a solution to these problems in the form of **Beatbox**—a portable, extensible and accessible computer simulation environment for computational biology of the heart.

Crucial to Beatbox's development will be openness to a broad user base. In particular, its success will depend on its ability to simultaneously address the needs of front-end users, for whom simple ease of interaction will be key; and developers, who require the power and flexibility to adapt the software to suit their needs. In this context, we extend the concept of openness beyond relaxed licensing and the availability of source code to the inclusion of a diverse community whose needs are evolving.

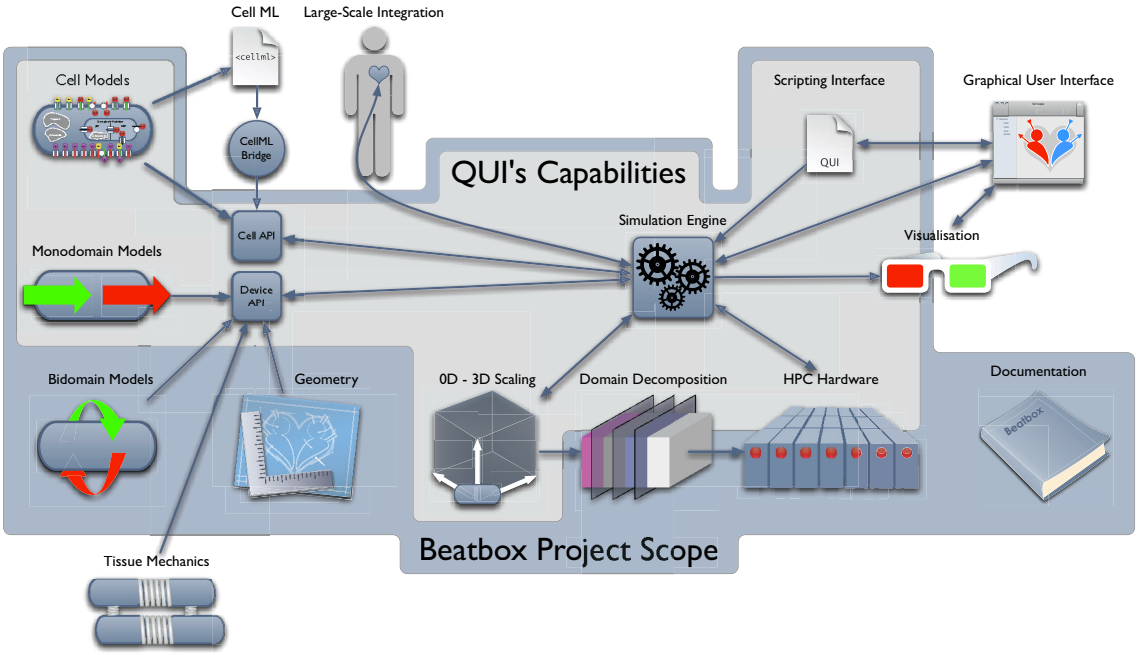


FIGURE 2: Beatbox Project Scope

5.1. QUI

The starting point for Beatbox's development is the QUI software package, originally developed at the Russian Academy of Sciences. Some of our early work has been to overcome the effects of QUI's own 'in-house' software development, namely its documentation, but QUI now provides much of the core functionality required of Beatbox. For example, QUI's simulation engine allows Beatbox to represent anything from a single-cell up to 3D tissues, selecting from 28 popular cell models. The project's work is expanding this palette to include bidomain models and realistic tissue geometry. Figure 2 shows the scope of the project related to the existing functionality of QUI.

5.2. Simulation Scripts

A Beatbox simulation is described using a high-level proprietary scripting language. Beatbox's in-built script interpreter, inherited from QUI, reads the script at runtime and builds the simulation accordingly. This breaks the edit-recompile-debug-repeat cycle found in much ad-hoc software and prevents the confusion and danger associated with exposing users to the program's source code. In practice, the script specifies the sequence of computational tasks to be run at each time-step of the simulation, such as computing diffusion or producing snapshot images of the medium. Aside from some core functionality, each task is performed by a modular component, called a *device*. Crucially, the user may call upon any of the available devices from the script, allowing the simulation to be quickly adapted. Such flexibility allows simple comparison of different models, parameter settings and output methods.

In a Beatbox script, devices can be called with a single line of code, with *key=value* pairs assigning parameter values. Standard device parameters include scope variables, which constrain the device to an area of the medium; and a boolean timing variable, used to indicate the time-steps in which the device should be run. Devices may specify further parameters, such as input or output files, values for which are requested from the script at runtime.

Scripts may be comprised of other scripts, allowing users to build simulations from even larger components. Runtime-interpreted scripts also offer simple exchange of simulation configurations between Beatbox users. Collaborators may quickly and easily adapt and transfer scripts, knowing

that their recipient will be able to easily understand and run them. In addition, Beatbox will also open simulations to scrutiny by peer review, which presents the opportunity to both accelerate and deepen the process.

5.3. Portability & HPC

Exchange of parameter settings is reliant on the portability of the core system. Beatbox extends QUI's existing portability to include a broad range of popular computing platforms. In addition, Beatbox enables users to harness HPC for their simulations with minimal input, handling the domain decomposition of the medium and the creation and co-ordination of threads as necessary.

Beatbox currently supports shared memory clusters via the popular OpenMP library (OpenMP Architecture Review Board 2002). The OpenMP parallelisation is applied to compute-intensive devices that iterate over every point in the medium. Loop iterations are divided between available threads, effectively dividing the medium between processors.

Support for distributed memory systems is provided via the Message Passing Interface (MPI) (Message Passing Interface Forum 2003). In the MPI implementation, each process is assigned an area of the medium. Each process runs its own instance of every device. Where a device needs only to access its local data, parallelisation is similar to that for OpenMP. For devices such as `diffusion`, where there is a need to read data from the boundaries of neighbouring regions, data from boundaries is exchanged between processes before computation begins.

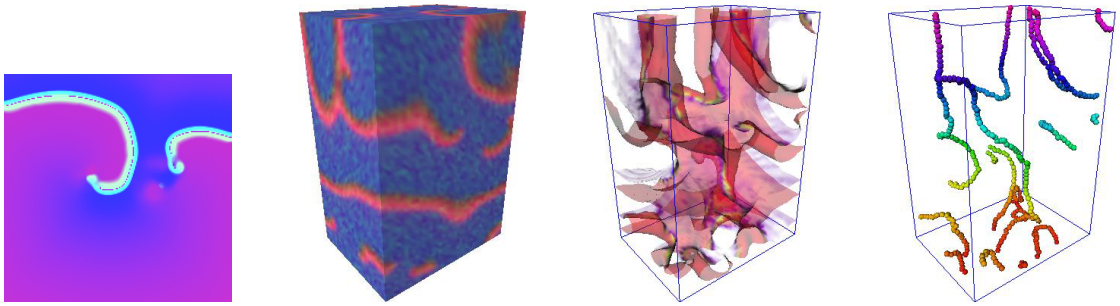
Future work will explore the combination of OpenMP and MPI for use in multicore hybrid architectures. When released, a canonical version of Beatbox will run on each of its supported platforms, irrespective of hardware configuration.

At present, each of Beatbox's devices must be parallelised with explicit calls to OpenMP or MPI library functions. This exposes device developers to the complexity of coding for HPC systems, which is ultimately against the principles of the project. This is being remedied by building core code behind the device API, simplifying the development of efficient devices and preventing unnecessary duplication of code.

5.4. Visualisation

Having completed a simulation, users can view their results using a suite of visualisation devices. For the simplest simulations, those that may be computed 'while you wait', X-Windows may be used to display 2D plots of specified parameters on screen. For the majority of simulations however, visualisation is constructed offline. The simplest of Beatbox's tools produces Portable Pixel Map images of the medium. In 2D, the device maps each point in the medium to a pixel in the image, while 3D images are represented as a stack of 2D slices. The colour at each pixel is determined by parameters in the simulation script, with each of the three (RGB) colour channels assigned a value from 0 to 1, computed from any value(s) stored in the medium. Snapshots may be taken at any frequency, as defined in the simulation script. From a sequence of output images, third party software may be used to produce a video of the simulation.

It is often desirable to simplify visualisation to highlight areas of interest. In particular, displaying wavefronts, or iso-lines of equal membrane potential can be useful aids to the interpretation of results. For the study of re-entrant waves, the centre of the spiral is a useful indicator and the ability to trace its *meander* across the medium is crucial to our understanding of re-entrant arrhythmias. In three dimensions, the *filament* of a re-entrant vortex is of particular interest. The filament is computed by discretising the vortex into a column or ring of stacked spiral waves. The core of each spiral wave is identified, and the filament is shown as the line through each of the cores in the vortex. Figure 3 shows a range of visualisation options possible using 3rd party visualisation software with QUI simulation data.



(a) 2D Plane (b) 3D Surfaces (c) Excitation fronts (d) Vortex filaments

FIGURE 3: **3D Visualisation of re-entrant vortices.** (a) direct output from QUI, (b,c,d) produced using third party software with simulation data produced by QUI. ©Vadim Biktashev, 2008. Used with permission.

Beatbox's suite of visualisation tools are in development to bring 3D functionality, like that used to produce figure 3 'on-board'. Section 6.2 discusses further desirable visualisation tools beyond the scope of the project.

5.5. Extensibility

Beatbox's longevity will depend on its ability to accommodate future advances in cardiac modelling, visualisation tools or computer hardware. To meet the future demands of its users, Beatbox's modular structure is extensible, allowing relatively inexperienced software developers to add functionality without exposure to the environment's inner workings.

New functionality is added as a Beatbox device, using the same API as in-built tools. Written in the C programming language, devices are essentially `struct`s, containing parameters, state variables and pointers to the functions that initialise, run and destroy the device. Shortcuts for common tasks are provided by a collection of macros. For example, the `ACCEPTP(<name>, <default>, <minimum>, <maximum>)` function macro 'pulls' a parameter value from the device's parameters in the simulation script.

Devices may access the simulation medium by specifying the cartesian coordinates of the cell. By convention, devices receive a `space` parameter from the script, defining the scope of the device. The limits provided by the `space` parameter can be used as loop bounds for devices that iterate over the medium. This is used in Beatbox's MPI parallelisation, to contain the devices in each process to their share of the medium.

The Beatbox device API obviates the need to write platform-specific code for any particular operating system or HPC configuration. Sequential devices may be included into parallelised versions of Beatbox without modification. At present, should a developer wish to parallelise device code, this can be done using standard OpenMP or MPI calls. By the time of its public release, Beatbox's API will include implicit parallelism, allowing new compute-intensive devices to efficiently harness HPC without complication to their development.

Once created, devices may be exchanged with other Beatbox users, allowing further collaboration and peer review. In this way, Beatbox users can combine their code with that of any other user; something that would prove prohibitively impractical without the common ground of an open simulation environment.

Underpinning openness for users and developers alike is comprehensive, approachable documentation. Without sufficiently detailed documentation, program code can quickly become outmoded or vestigial. Releases of Beatbox will be accompanied by comprehensive documentation of both the user interface, i.e. the scripting language, and the APIs against which new devices may be written.

6. FURTHER WORK

Our aim is to release Beatbox, as described above, within the course of this three year project. What follows is a discussion of those components outwith the scope of the project, but worthy of inclusion at a later stage.

6.1. CellML Bridge

CellML (Hedley et al. 2000) is an XML-based language used to represent models of cell kinetics. Increasingly, CellML descriptions are being used to accompany publications of models, creating members of the community opportunity to rapidly generate simulation code, rather than attempt to reproduce it from published equations. Beatbox would benefit from a software bridge capable of parsing CellML in order to produce a native Beatbox cell model.

6.2. 4D Visualisation

At the opposite end of the simulation workflow, the analysis and interpretation of simulation results relies upon the clarity of their presentation. While Beatbox's visualisation tools offer adequate detail and scope, they would benefit from the ability to interactively manipulate the visualisation; step through the x, y or z-stack; rotate the medium along any axis; and exact fluid control over the speed and direction of playback.

Beyond front-end users, the detailed and intuitive presentation of simulation results can aid communication with third parties. For example, simulation results may elucidate the processes underlying arrhythmias to clinicians, or assist in providing a tangible measure of progress for project stakeholders.

In practice, a 4D visualisation tool for Beatbox need not be implemented in full. Export to formats supported by third-party applications may allow the community to benefit from progress in other fields.

6.3. GUI

Simulation scripts provide a useful intermediary for collaborating researchers and allow simple repetition of simulations for the purpose of peer review. However, the majority of our interaction with computer software today benefits from the use of graphical user interfaces (GUIs) and consideration should be given to the ways in which GUIs might be harnessed to improve ease of use for less technically-oriented researchers. A short-term step may be to provide a GUI with which scripts may be edited interactively, ensuring the correctness of the script and making its effects more immediately apparent. A number of commercial software tools from other fields offer a visual approach to programming tasks, e.g. Adobe Flash (Adobe Systems Incorporated 2008), Cycling '74 Max/MSP (Cycling '74 2008) and Apple Automator (Apple Inc. 2008).

6.4. Self-Documenting Modules

Given that much of the code in use may be community developed, we should consider how this might be documented and how any such documentation will be distributed. One particularly attractive idea is that modules might be self-describing, allowing their functions, parameters and provenance to be read from the module itself. For example, a GUI tool may allow the user to assemble a simulation from the available collection of modules and set their parameters. By reading each module's self-description, the GUI tool can inform the user of the purpose of each parameter and how the module will interact with others.

7. CONCLUSIONS

Beatbox provides an exciting opportunity to raise the standard of simulation software, facilitating improved use of HPC, together with exchange of simulation components, parameters and results.

Beatbox will fill the gap left by ad-hoc software development and foster a culture of open information exchange.

Our aims regarding openness, communication and collaboration are in line with those of the International Union of Physical Sciences Physiome project (Hunter et al. 2006), whose work seeks to provide a framework for multiscale modelling of physiological processes. Where the Physiome project's markup languages offer a means of communicating models, Beatbox provides the tools to build simulations from those models that will run efficiently and ubiquitously. Open and inclusive software design, together with effective information interchange, will improve intra- and inter-disciplinary collaboration, communication and peer review.

8. ACKNOWLEDGEMENTS

We thank Profs. Vadim N. Biktashev and Trevor J.M. Bench-Capon for their assistance. Financial support of the Beatbox project is from The Engineering and Physical Sciences Research Council, UK, through a DTA studentship for R. M^{c}Farlane, together with The University of Liverpool Computer Science Department. Original (OpenMP) parallelisation of QUI was funded under The University of Liverpool Research Development Fund project 4431.

References

Adobe Systems Incorporated (2008). *Flash CS3*. URL: `http://www.adobe.com/products/flash/`.

Apple Inc. (2008). *Automator*. URL: `http://www.apple.com/macosx/features/300.html#automator`.

Biktashev, V.N. and Karpov (Unpublished). "QUI Software Package".

British Heart Foundation Health Promotion Research Group (2008). "European Cardiovascular Disease Statistics". In: *BHF Coronary heart disease statistics at www.heartstats.org*. URL: `http://www.heartstats.org/temp/ESspweb08spchapter.1.pdf`.

Clancy, Colleen E and Yoram Rudy (1999). "Linking a genetic defect to its cellular phenotype in a cardiac arrhythmia". In: *Nature* 400.5 August. Pp. 566–569.

Courtemanche, Marc, Rafael J Ramirez, and Stanley Nattel (1998). "Ionic mechanisms underlying human atrial action potential properties: insights from a mathematical model". In: *Am J Physiol Heart Circ Physiol* 275.1. H301–321.

Cycling '74 (2008). *Max 5*. URL: `http://www.cycling74.com/products/max5`.

FitzHugh, R (1961). "Impulses and Physiological States in Theoretical Models of Nerve Membrane". In: *Biophysical Journal* (Jan. 1961).

Hedley, W.J. et al. (2000). "XML Languages for Describing Biological Models". In: *Proceedings of the Physiological Society of New Zealand*. Vol. 19.

Henriquez, C and A Papazoglou (1996). "Using computer models to understand the roles of tissue structureand membrane dynamics in . . . ". In: *Proceedings of the IEEE* (Jan. 1996).

Hsu, E et al. (1998). "Magnetic resonance myocardial fiber-orientation mapping with direct histological correlation". In: *American Journal of Physiology- Heart and Circulatory . . .* (Jan. 1998).

Hunter, P et al. (2006). "Multiscale Modeling: Physiome Project Standards, Tools, and Databases". In: *Computer* (Jan. 2006).

LeGrice, I (2001). "The architecture of the heart: a data-based model". In: *Philosophical Transactions: Mathematical* (Jan. 2001).

Luo, Ching-Hsing and Y Rudy (1994). "A dynamic model of the cardiac ventricular action potential. I. Simulations of ionic currents and concentration changes". In: *Circulation Research* 74.6 (June 1994). Pp. 1071–1096.

Message Passing Interface Forum (2003). *MPI: A Message-Passing Interface Standard*.

Noble, D (2002*a*). "Modeling the Heart—from Genes to Cells to the Whole Organ". In: *Science* (Jan. 2002).

— (2002*b*). "Unraveling the genetics and mechanisms of cardiac arrhythmia". In: *Proceedings of the National Academy of Sciences* (Jan. 2002).

OpenMP Architecture Review Board (2002). *OpenMP C and C++ Application Program Interface*.

Panfilov, AV, R Keldermann, and M Nash (2007). “Drift and breakup of spiral waves in reaction-diffusion-mechanics systems”. In: *Proceedings of the National Academy of Sciences* (Jan. 2007).
Sundnes, Joakim et al. (2006). *Computing the Electrical Activity in the Heart*. 1st ed. Monographs in Science and Engineering 1. Berlin / Heidelberg: Springer Verlag.
Veldkamp, M, P Viswanathan, and C Bezzina (2000). “Two Distinct Congenital Arrhythmias Evoked by a Multidysfunctional Na+ Channel”. In: *Circulation Research* (Jan. 2000).
Wiener, N. and A. Rosenblueth (1946). “The mathematical formulation of the problem of conduction of impulses in a network of connected excitable elements, specifically in cardiac muscle.”. Russian. English original. In: *Kibern. Sb.* 3. Pp. 7–56.