

International Conference on Computational Science, ICCS 2013

## Connecting models to data in multiscale multicellular tissue simulations

Jonathan Cooper<sup>a,\*</sup>, James Osborne<sup>a</sup><sup>a</sup>*University of Oxford, Department of Computer Science, Parks Road, Oxford, OX1 3QD, UK*

---

### Abstract

System level biological behaviour typically arises from highly dynamic, strongly nonlinear, tightly coupled interactions between component processes occurring across multiple space and time scales. The interdependent nature of these processes often makes it difficult to apply standard mathematical techniques to separate out the scales, uncouple the physical processes or average over contributions from discrete components. To make rapid progress we need to address interoperability challenges: to build integrated models from reusable components, and to relate simulation results to experimental data both for parameter fitting and model analysis. In this paper we describe how work we have done to address these issues in the domain of cardiac electrophysiology can be applied in a completely different field: multicellular models of intestinal crypts, with cells treated as discrete entities, and the sub-cellular, cellular, and tissue scales interacting. In this application the model and simulation are intertwined in software, with no suitable markup language model representation. Different modelling paradigms are available for each of the scales, and comparing their predictions is of particular interest. We use our concept of ‘functional curation’ to separate the experimental protocols applied to models from the model descriptions themselves, allowing easier comparison of model behaviour with experimental data. We also describe the use of ontological annotation for providing semantically rich model interfaces, facilitating coupling models to each other and to protocol descriptions. Finally, we show how these uses of semantic annotation and markup languages may be mixed incrementally with legacy code. This work suggests that the ideas we have developed have the potential to be useful across computational science, and we discuss these wider implications.

**Keywords:** Functional curation, Chaste, SED-ML

---

### 1. Introduction

Two related challenges in building mathematical models of biological systems stem from the complexity of the systems being modelled. Few biological systems are small and self-contained. Interactions occur across multiple levels of organisation and are usually intricate and nonlinear, making an intuitive grasp of their function hard to obtain. Often, many component parts of a system must be modelled and the models tightly coupled in order to gain insight into overall behaviour. Building reusable component models which can be integrated in such a way is a significant challenge. However, models are only useful if they provide insight into or predictions of experimentally observed behaviour, and thus an equally important challenge is to relate the results of simulating such models to experimental data. This must be done both in developing models, for instance by fitting model parameters, and also in model evaluation, for example in validation, verification and uncertainty quantification [1].

---

\*Corresponding author. Tel.: +44-1865-610671 ; fax: +44-1865-273839 .

E-mail address: [jonathan.cooper@cs.ox.ac.uk](mailto:jonathan.cooper@cs.ox.ac.uk).

We have been creating tools as part of the Chaste simulation environment [2] to tackle these issues as they arise in the context of cardiac electrophysiology. In this domain, cellular level models describing the activity of muscle cells must be coupled to a tissue level model describing the spatial variation in electrical potential. Many alternative cellular level models exist, and in [3] we describe how these may be coupled to our tissue model largely automatically. We are also developing a framework for performing ‘functional curation’ [4] of such cellular models, in order to evaluate and compare their behaviour under a wide range of experimental scenarios.

We next, in Section 2, describe how this previous work relates to the two modelling challenges outlined above. In Section 3 we outline the new application domain of multicellular modelling and its challenges, and then present a case study of how our techniques (with minor adaptations) can fruitfully be applied. We discuss the wider implications and research questions arising from our work in Section 4, and conclude in Section 5.

## 2. Background

In this section we describe our previous work, in the domain of cardiac electrophysiology, on addressing the twin challenges of building reusable component models which can be integrated together, and relating the results of simulating such models to experimental data.

### 2.1. Building interoperable model components

To tackle the first challenge, encoding models in standard markup languages (e.g. CellML [5] or SBML [6]) allows for easy reuse of model equations. However, in [3] we saw that one needs to go beyond a standards-based representation of the model mathematics, and include additional semantic information to facilitate determining a suitable model interface for the particular reuse scenario. The key elements are the identification of the model entities which should be connected, and the resolution of inconsistencies between component models.

When a biological process is modelled through computational simulation, the first step is to develop a ‘biological model’ of the process—a qualitative verbal or graphical description as might be expressed by a biologist. Converting this biological model into an interrogatable quantitative simulation requires various steps. We must first generate a mathematical model which assigns quantitative meaning to biological concepts. Next appropriate numerical methods (specifying resolutions of the approximations) must be selected, and then implemented in a programming language or computational framework. With many, often novel, variations in how these steps may be performed being available, a raft of different implementations of the biological model can be produced.

Ideally, coupling should be performed at the level of the ‘biological model’, and the transformation steps automated so as to account for the implications of these new connections. Typically, however, the biological model is implicit, and only the mathematical, numerical or computational model is available for reuse. In [3] we thus demonstrated how a mathematical model may be analysed by mostly automatic tools to infer sufficient biological information for a specific coupling scenario. The easier problem to address is the identification of model entities in different component models which represent the same biological entity, and hence should be connected when the models are coupled. Semantic metadata encoded using the Resource Description Framework [7] may be used to annotate model entities with ontology terms uniquely defining the biological entity. This will, at least where agreement on the ontology to use is found, avoid conflicts arising from variations in naming conventions. It also makes explicit the biological meaning of model entities, aiding in model comprehension.

A further problem arising in many model coupling exercises is the existence of incompatibilities between the component models. These primarily arise from differences in the ways in which constituent models have been transformed into their computational forms. Three categories of such incompatibilities were defined by Terkildsen *et al.* [8] as unit, structural, and parameter inconsistencies. The easiest case is *unit inconsistencies*, where scaling between quantities measured in different units of the same dimension may be performed automatically [9]. *Structural inconsistencies* refer to differences in how the biological system is represented by the mathematical equations. These often manifest as the same quantity measured in units of differing dimensions. In our functional curation framework, users may define conversion rules to be applied between such quantities, which can utilise biological information encoded within the model in order to map between different representations. Where the cue provided by a unit incompatibility is not present, the modeller is required to intervene manually. *Parameter inconsistencies* arise when component models use different estimates for the same biological quantity. Where such

parameters are annotated with biological information, tools at least may alert the user, but choosing what value is most appropriate for the combined model remains a manual task.

## 2.2. Interoperability between models and experimental data

While the above concepts can make it technically easier to reuse a model in a new context, they do not imply that doing so is in any way biologically realistic. Models are usually developed to represent particular experimental setups, with associated parameter values and initial conditions, and may give unexpected results when used outside that regime. The scientific questions for which the model is relevant must therefore be determined, and our work on *functional curation* [4] aims to address this issue. Effective reuse requires an understanding of a model's functional capabilities, which must be confirmed by simulation and comparison against experimental data.

Much of the work required in interfacing component models may also be usefully applied when interfacing models to experimental data. More fundamentally, however, it is necessary to be able to simulate from the model exactly the protocol used to generate the data, including pre- and post-processing, so that the results are directly comparable (c.f. [10]). A particular difficulty faced when attempting to characterise a model against multiple experiments is that many mathematical models, whether available in standard formats or especially provided as monolithic software, contain not just the model structure (i.e. the mathematical equations describing the biological function), but also a precise applied experimental protocol, including the parameter values and initial conditions associated with that protocol. It is therefore necessary to 'unpick' the 'hardwiring' of the protocol and the precise parameterisation, so that only the model structure remains. This is often a complex and error-prone procedure.

Recently a new approach has emerged which has the potential to overcome this very limiting problem by separating an experimental protocol from the model structure. The mathematical model then corresponds much more directly to the actual biological system, in that it contains representations of all of the necessary functional biological components, but no experimental intervention. The user must then embed this model within a simulation protocol, which is the direct *in silico* equivalent of a corresponding wet-lab experimental protocol, providing all the information necessary to perform an experiment that either measures some property of the system directly or (more commonly) perturbs the system and measures its response. Composition of such models is then directly analogous to including additional features of the biology that is being modelled, and is completely decoupled from the manner in which the model is interrogated. This brings us closer to the goal of coupling at the level of the biological model. It also reduces the impact of parameter inconsistencies, since parameterisation for a given scenario should be part of the protocol, not the model.

The emerging community standard for encoding and sharing these simulation protocols is called SED-ML (Simulation Experiment Description Markup Language [11]), and we are working to incorporate extensions from our functional curation tools into future versions of this standard. Our additional concepts include an interface layer between the protocol and the model, which uses the techniques described in Section 2.1 to ensure that model references from the protocol are well defined and unambiguous. This facilitates the application of a single protocol to a range of models (e.g. to compare different hypotheses). Another key feature is the use of regular  $n$ -dimensional arrays as the basic data type (beyond just 1d vectors or 2d matrices). These allow results from simulations with arbitrary levels of nested loops to be represented (at least when model outputs are of fixed shape) and manipulated by the post-processing constructs. The latter aim to balance expressivity with ease of implementation, providing the maximal utility at least expense to tool developers required to support an exchange standard. Finally, note that a protocol itself has inputs and outputs. Protocols may be imported, allowing libraries of functionality to be reused or general experimental setups specialised to particular scenarios. Protocols may also be considered as a model, allowing them to be nested within further simulation loops.

## 3. Application to multicellular models

In this section we describe how the protocol and interfacing concepts described above, developed originally for single cell cardiac electrophysiology applications, may be applied in a different modelling field, in order to automate partially a simulation experiment of the type that is routinely performed manually. We begin by describing the application domain (Section 3.1) and specific scenario (Section 3.2). In Section 3.3 we describe how our functional curation system has been extended for use in this new domain, and show the full protocols used in this case study. Finally, the results of running the protocols are given in Section 3.4.

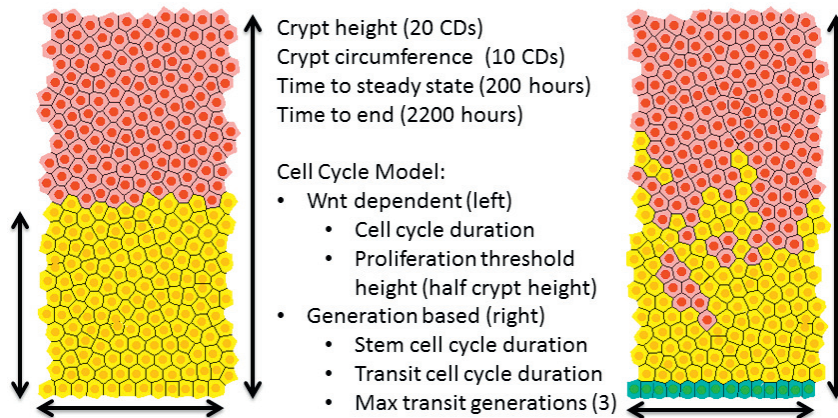


Fig. 1: Setup of the crypt model at quasi-steady state. The left figure shows a crypt using a Wnt-dependent cell cycle model; and the right uses a stochastic generation-based cell cycle model. Parameters that vary in the model are given in the centre and where appropriate default parameter values, which are used, are given in brackets. Distances are given in terms of nondimensional cell diameters (CDs).

### 3.1. Multicellular modelling of intestinal crypts

The intestinal epithelium is the most rapidly regenerating surface in the human body. The renewal of this epithelium is coordinated by millions of test tube shaped glands known as colorectal crypts which are lined with a layer of cells, and the process is governed by balanced proliferation, migration and death within the crypt. Cells proliferate in the lower part of the crypt then subsequently migrate towards the intestinal epithelium. The colorectal crypt forms a model system for simulation of tissue growth and development. Moreover it is the site for the onset of colorectal cancer. Therefore understanding how it functions, and how this function goes wrong, is of key interest [12, 13]. For a recent review covering modelling in the crypt see [14].

Here we study a model of the colorectal crypt implemented within the the cell-based side of the Chaste framework. Cell-based Chaste represents a state of the art multiscale multicellular modelling framework which allows users to develop their own multicellular simulations. Utilising the natural structural unit of the cell, the framework consists of three main scales: the tissue level (macro-scale); the cell level (meso-scale); and the sub-cellular level (micro-scale), with interactions occurring between all scales. The cell level is central to the framework and cells are modelled as discrete mobile interacting entities using one of a number of possible modelling paradigms, including lattice based models (cellular automata and cellular Potts) and off-lattice models (cell centre and vertex based representations). The sub-cellular level concerns numerous metabolic and biochemical processes represented by interaction networks, rendered stochastically or into ordinary differential equations. The outputs from such systems influence the behaviour of the cell level, affecting properties such as adhesion, and also influencing cell mitosis and apoptosis. Tissue level behaviour is represented by field equations for nutrient or messenger chemical concentrations, with cells functioning as sinks and sources. This modular approach enables more realistic behaviour to be considered at each scale [15].

Typical experiments performed with cell-based Chaste include parameter scans over cell-level properties to see how these influence the evolution of the population of cells as a whole, and post-processing the resulting cell-level data (e.g. protein concentrations, cell position and type) to generate population-level statistics (e.g. distributions of cell types, global birth and death rates) which are required to compare simulations to each other and to experimental data (see e.g. [16, 15, 12, 17]). As an illustrative virtual experiment, we compare three models for cell proliferation to see how they are affected by varying the crypt geometry.

### 3.2. The models

We utilise a cell centre based model for the crypt described in [16] and [15]. Figure 1 shows the setup of the model and all parameters that we may vary. Full details can be found in [15], and all parameters used here

are the same unless otherwise indicated in Figure 1. We compare three different component models of the cell cycle, described below. For all three models proliferating cells divide stochastically, where the cell cycle duration (CCD), measured in hours, is drawn from a uniform distribution.

**Uniform Wnt:** Here, whether cells may divide is determined by the concentration of the Wnt signalling factor, which decreases up the crypt, so that cells no longer proliferate in the top half of the crypt. While proliferating the CCD is drawn from a Uniform(10,14) distribution. This simple model is found in [15].

**Variable Wnt:** This is a generalisation of the above model and consists of two modifications. Firstly, instead of the proliferative state of the cell depending on its current position, it is set depending on the position when the cell is born (therefore cells will only differentiate on division). The threshold level for this to occur is still taken to be halfway up the crypt. Secondly the CCD varies with position and is drawn from a Uniform( $m_t - 2, m_t + 2$ ) distribution, where  $m_t = 24 \times (1 - y)$  is the mean CCD and  $y$  is the relative position of the parent cell up the crypt upon division. Therefore cells at the base of the crypt will have a mean CCD of 24 hours and cells in the middle of the crypt will have a mean CCD of 12 hours.

**Stochastic Generation Based:** Here we have two proliferative cell types: stem and transit. Stem cells reside at the base of the crypt and divide asymmetrically to produce a stem and a transit cell. Transit cells divide for a predetermined number of generations (here chosen to be 3), before terminally differentiating. The CCD for stem and transit cells are drawn from Uniform(22,26) and Uniform(10,14) distributions respectively. Details of this model can be found in [16].

### 3.3. The simulation protocols

The use of functional curation allows experiments on models such as those described above to be run in a more intuitive, general and robust fashion than has been possible. Previously experiments in cell-based Chaste had to be defined by writing C++ code to set all parameters expressly and perform post processing. Functional curation allows a single experiment to be applied to any computational representation of the biological system. This will enable the determination of which properties are model specific and which are actually predictions of the biological system, since it is not generally clear which modelling paradigm is appropriate for each application.

One extension to [4] has been the creation of a textual syntax for our protocol language. This both makes it easier for users to write protocols (as compared to writing raw XML), and provides a compact readable description, with the potential to be incorporated in publications as we have done here. Our case study consists of two protocols, the first of which is shown in Listing 1. This is responsible for running a single cell-based Chaste simulation and performing some post-processing on the raw division data which it outputs. This protocol is then nested within the parameter sweep shown in Listing 2. Both protocols are commented to explain individual sections; we remark on a few key features here.

Unlike the cardiac applications in our earlier paper, where models consisted of ordinary differential equations and could be manipulated extensively by the protocol, here a model is in essence an executable program which must be run in its entirety. We have thus needed to create a new implementation of the interface layer, allowing model parameters to be set (in appropriate units) from a protocol, prior to execution of the model. Note, however, that supporting this new application field required only one change to the protocol language—the new **oneStep** simulation type, which encapsulates running a single execution of the model—a change that had already been considered for SED-ML in other contexts. The more generic aspects of functional curation still apply: the model has certain parameters which may be set prior to execution, the execution may be nested within further simulation loops, and post-processing of the results may be performed. For this example we specify only the simulation duration and crypt height; other parameters are left at their default values indicated in Figure 1.

Note the ability to reuse (parts of) protocols, particularly for post-processing operations. Standard utility functions such as `std::After` are imported; other functions are defined locally to this protocol. Many MathML operators may be used to perform calculations; the MathML: prefix gives access to those which don't have special syntax. Extensions for  $n$ -dimensional array operations are also demonstrated, for instance: creating an array using a *comprehension* as in lines 17 and 35; extracting sub-arrays as well as single values when indexing arrays (e.g. line 40), and applying functions element-wise to arrays using **map** as in lines 37 and 46.



The inner protocol defines both inputs and outputs. This allows it to be considered as a kind of model by the outer protocol. This parameter sweep defines a single loop varying the `crypt_height` variable, which is used to set the corresponding input of the inner protocol for each run. A subset of the available outputs of the inner protocol are selected as being of interest; these must have the same shape on each inner run in order to maintain the regular array data model. Finally, the outputs of the outer protocol, and default line plots which will be generated automatically by the system, are declared.

Listing 1: Inner protocol for post-processing a single parameterised simulation.

---

```

# The 'ontology' to use for referencing model variables
namespace cellbased = 'https://chaste.cs.ox.ac.uk/nss/cellbased/0.1#'
inputs { # Protocol inputs
    num_boxes = 10 # The number of boxes to use in the location histogram
    crypt_height = 20 # The height of the crypt (in nominal cell diameters)
    end_time = 2200 # The simulation end time (hours)
    # The time at which the system is assumed to have reached quasi steady state (hours).
    # We ignore division events occurring before this point.
    steady_state_time = 200
}
# Import the standard library of post-processing operations, using a relative path.
# Functions from this library may then be used by prefixing their names with 'std:'.
import std = '../.../FunctionalCuration/src/proto/library/BasicLibrary.xml'
library { # Define some extra utility functions
    def InBox(loc, boxLow, boxHigh) { return loc >= boxLow && loc < boxHigh }
    # Extend an array by copying it a given number of times along a newly added dimension
    Stretch = lambda array, length, dim: [array for dim$ i in 0:length]
}
units { # Units definitions for this protocol
    hours = 3600 second
    lengthUnits = 10 micro metre "Nominal cell diameters"
}
tasks { # The raw simulations to perform
    # Just run the cell-based simulation as-is, setting a few parameters at the start
    simulation sim = oneStep {
        modifiers {
            at start set cellbased:end_time = end_time
            at start set cellbased:crypt_length = crypt_height
            at start set cellbased:cells_up = MathML:ceiling(crypt_height * 2 / MathML:root(3))
        }
    }
}
post-processing {
    box_size = crypt_height / num_boxes # y coordinates start at zero
    box_lows = [i*box_size for i in 0:num_boxes]
    box_highs = [(i+1)*box_size for i in 0:num_boxes]
    centres = map(lambda a, b: (a+b)/2, box_lows, box_highs)
    # The main simulation output is the 2d array of division data, with 4 columns:
    # time, x, y, age. We extract division y coordinates for events after the given time.
    locations = std:After(sim:divisions[1$2], sim:divisions[1$0], steady_state_time)
    num_divisions = locations.SHAPE[0]
    # Figure out which histogram box each cell division occurred in, and count them up
    locations_ext = Stretch(locations, num_boxes, 0) # Make all the _ext arrays the same
    box_lows_ext = Stretch(box_lows, num_divisions, 1) # shape: [num_boxes, num_divisions]
    box_highs_ext = Stretch(box_highs, num_divisions, 1)
    in_box_pattern = map(InBox, locations_ext, box_lows_ext, box_highs_ext)
    freqs = std:RemoveDim(std:Sum(in_box_pattern), 1) # Shape [num_boxes]
    assert std:RemoveDim(std:Sum(freqs), 0) == num_divisions # Sanity check
}
50 outputs {
    divisions = sim:divisions "Raw division data" # Shape [num_divisions, 4]
    freqs units dimensionless "Number of divisions per box" # Shape [num_boxes]
    centres units lengthUnits "Box centres" # Shape [num_boxes]
}
55 plots {

```

```

    plot 'Cell division locations' { freqs against centres }
}

```

Listing 2: Outer parameter sweep, which runs the protocol in Listing 1 for a range of crypt heights.

```

1  inputs {
    num_boxes = 10                # The number of boxes to use in the location histogram
    heights = [10, 15, 20, 25, 30] # The crypt heights to sweep over
}
import std = '../.../FunctionalCuration/src/proto/library/BasicLibrary.xml'
6  units { percent = dimensionless "%" }
    tasks {
        simulation sweep = nested {
            range crypt_height units lengthUnits vector heights
            nests protocol 'CryptProliferation.txt' {
11             num_boxes = num_boxes          # Pass through
                crypt_height = crypt_height    # Set crypt height for this iteration
                # Output of interest, with shape [num_boxes] for a single protocol run
                select output freqs
            }? # Turn on debug tracing, so the outputs of each run are saved separately
16        }
    }
    post-processing {
        # Compute box centres as % of crypt height, for plotting all crypts on the same axes
        centres_percent = [(100/num_boxes)*(box_num+0.5) for box_num in 0:num_boxes]
21        # Normalise division counts for easier comparison
        total_divisions = std:Stretch(std:Sum(sweep:freqs, 1), num_boxes, 1)
        norm_freqs = map(lambda n, tot: n/tot*100, sweep:freqs, total_divisions)
    }
    outputs {
26        freqs = sweep:freqs      units dimensionless "Number of divisions per box"
        norm_freqs                units percent      "Percentage of divisions per box"
        centres_percent            units percent      "Percentage height up the crypt"
        heights                    units dimensionless "Crypt height" # Note: fake units for display
    }
31  plots {
    plot 'Cell division locations' { norm_freqs against centres_percent key heights }
}

```

### 3.4. Results

Figure 2 shows the results of running the protocol in Listing 2 on crypt models featuring the three different cell-cycle models presented in Section 3.2; the models are otherwise identical. The graphs are produced entirely automatically by our system, and all code and data required to reproduce them is available from our website.<sup>1</sup>

By comparing Figure 2(a)-(c) to (d) we can see that the results for the Variable Wnt model are most similar to the experimental data. A closer fit could be obtained by using the framework to perform a larger parameter sweep over more of the model parameters (CCDs for example). By looking at individual figures we can see the effect that varying crypt height has on the distribution of division events. From Figure 2(a) we see that away from the base of the crypt the distribution of division events is relatively uniform and independent of crypt height. Crowding at the crypt base leads to more divisions there since the model does not contain any form of contact inhibition. The Variable Wnt model is affected most by varying height. Figure 2(b) shows that as the crypt height is increased there is a greater proportion of division events in the middle region of the crypt. This is due to the increased number of cells in the lower region of the crypt producing a larger upward force on cells above them, meaning that a cell which divides in the bottom half of the crypt will be able to divide again when it has moved further up the crypt. For the generation-based model the absolute distribution of division events is largely independent of crypt height, since divisions are based on generation count not position. However as we plot the relative distribution of divisions in Figure 2(c) we see that as the height is increased the distribution becomes more skewed towards the base of the crypt, as you would expect from considering the relative heights.

<sup>1</sup><https://chaste.cs.ox.ac.uk/trac/wiki/PaperTutorials/Wisc2013>

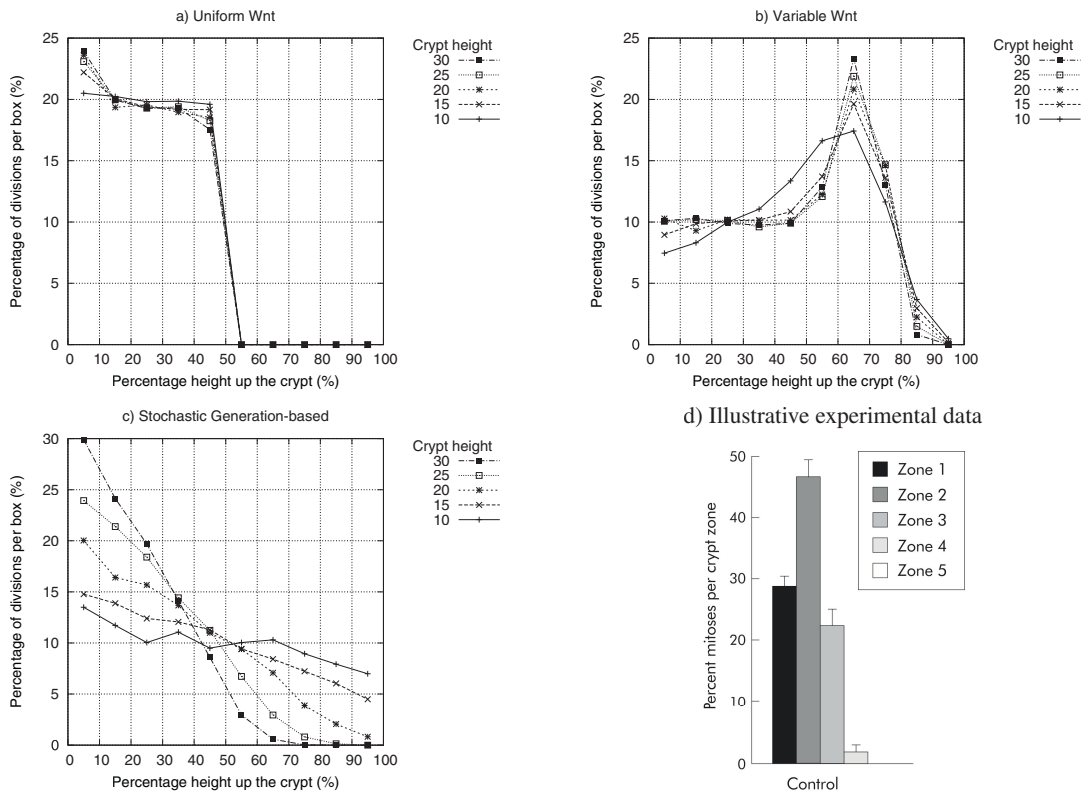


Fig. 2: (a)-(c) Distributions of cell division events with different cell cycle models. (d) Illustrative experimental data on the position of division events within a healthy crypt. The five zones correspond to dividing the crypt vertically into five equally sized segments with zone 1 at the base. Modified with permission from [18].

## 4. Discussion

We have shown in the case study above that our ‘functional curation’ concept can fruitfully be applied to modelling fields beyond cardiac electrophysiology, and that many of the features present in our language for encoding simulation experiment protocols have utility across application domains. This provides evidence for our conviction that our protocol language constructs represent a good abstraction for describing the generic (as opposed to unavoidably model-specific) aspects of simulation definition, data analysis, post-processing, etc. However, new applications also provide new challenges for protocol description which may require a shift in approach, and there is further scope for the model interfacing ideas from our earlier work to be leveraged.

### 4.1. Coupling models in multicellular tissues

Models in cell-based Chaste are implemented as C++ source code, and a complete model must be constructed by instantiating classes representing the various submodels and connecting the resulting objects. Our work has improved upon this by allowing aspects of the experiment being run to be extracted into a protocol definition, thus increasing separation of concerns. However, the model itself must still be constructed as a monolithic entity.

Portions of the cell-based models, notably the subcellular reaction networks and cell-cycle models, can be described with SBML. We are thus investigating using SBML for these submodels, using ontological annotations and units conversions to interface them with each other and with the higher-level submodels. The situation is more complex than the cardiac case (where the cell/tissue interface is essentially the same for all models) since the models in question are much more heterogeneous. A cell may potentially contain many different signalling networks for instance, represented by different SBML descriptions, which may interact wherever they contain a



common protein or other molecule in the same cellular compartment. Many different chemicals may also be used to communicate externally to the cell, and hence form part of the interface both to the tissue level model, and with neighbouring cells. The interfacing framework we have described can handle many of these complexities, and will be more flexible than at present, especially in being more scalable as new submodels are developed.

For the multicellular aspects of the models, no markup language standard currently exists, although work is progressing on SBML extension packages that may be suitable for some cases. Again, the wide variety of modelling approaches makes defining a set of standards that can encode all the variations a huge challenge. Indeed, it is likely that there will always be a need for fully flexible general purpose programming languages in defining some kinds of models, since novel models may require features not thought of in any standard.

However different component models are defined, specifying the particular combination and connections to use in a given experiment from a protocol description requires further work, especially to do so in a concise and user-friendly manner. For example, we want to be able to write a protocol that specifies “use a cell-centre-based mechanics model, with these cell types in these geometric regions, using these subcellular models in those regions” and be able to specify parameter values for each submodel in different regions. This will require at the least adding some concept of space to the protocol language, and graphical tools for user-friendly problem definition. However, the use of ontology terms for specifying parameters and matching up submodel interfaces, with automatic units conversion where appropriate, will still be relevant.

One open question is how best to map parameters defined at the level of the biological model to relevant quantities in the mathematical or computational model. As a concrete example, one of the parameters to the Wnt-based cell-cycle models in the case study above is the threshold Wnt signal level below which a cell will no longer divide. There is no direct analogue for this parameter in the third cell-cycle model. However, since the Wnt signal is specified (in another submodel) to reduce linearly with height up the crypt, the threshold level can be converted to a threshold height. Assuming that each successive generation of cells lies approximately one cell diameter further up the crypt, this height can then be converted to a generation number at which cells will no longer divide, which is the parameter that has the same behavioural effect. Performing such an analysis automatically is non-trivial; it is likely that users will need to specify ‘mini models’ which encode these transformations. Given a library of such transformer models, tools may be able to select a suitable transformation (or chain of transformations) automatically based on the submodels being coupled and the parameters being set from the protocol.

#### 4.2. *Extensions and alternatives for describing protocols*

Other implications for protocol languages arise from this study. Notably, the restriction of the core datatype to *regular*  $n$ -dimensional arrays poses difficulties for simulations involving cell birth and death, since many of the natural outputs will therefore not have the same shape throughout a timecourse simulation. In some cases it is possible to work around this limitation through judicious use of post-processing within a nested protocol, as we have done above. However this is often not possible, and so support for ‘ragged arrays’ will be required, wherein sub-arrays do not have to have the same shape (for example, a vector of vectors of lengths 2, 3, 5, and 11 would be permitted). Such a structure is supported by scientific data formats such as HDF5 [19], and hence is not necessarily an unreasonable obstacle for implementations of a protocol language.

Another feature that is required if protocol descriptions are to play a central role in model development is the ability to specify how to fit models to experimental data. This is a natural extension, since a simulation protocol should correspond to the experimental protocol used in obtaining data [10], and so the protocol outputs should correspond directly to the experimental data, and hence be well suited for being incorporated in an objective function. Complexities arise due to the wide variety of parameter fitting approaches, and the difficulty of interfacing to experimental data formats. These issues are currently being discussed by the SED-ML community, and others are welcome to contribute on the mailing list.<sup>2</sup>

## 5. Conclusions

The above case study highlights two themes which we believe are central to increasing the utility of computational modelling through improved interoperability. The first is the use of domain specific languages to extract

<sup>2</sup><https://lists.sourceforge.net/lists/listinfo/sed-ml-discuss/>

high level descriptions of both models and simulation protocols from computational codes. This separation of concerns can provide a clearer description of the essential concepts, without being cluttered with implementation details. It also enables assumptions made in the process of converting a biological hypothesis to a computational model to be made explicit. The second theme is the need for semantically rich interfaces between component models, protocols, and data, in order to build, validate, and compare multiscale models. Connections made at the level of the ‘biological model’ can avoid conflicts from inconsistencies arising due to differences in the ways in which different components have been constructed as mathematical or computational models. While in this paper we have considered only examples of physiological modelling, we believe these general principles, and indeed the broader framework of functional curation, can be applied much more widely within computational science.

However, it is clear that there is much work still needed on these aspects and others. In particular, implementing new domain specific languages for areas where no standards yet exist is considerably more challenging than coding up an ad-hoc solution, especially for research scientists with little formal training in computer science. The flexibility of general purpose languages will always be needed to support novel research. Nevertheless, there is a need both for improved tools to make separating aspects of complex models easier, and training for scientists to think about computational modelling in a different way. We hope that our work will help with the first problem, while efforts such as Software Carpentry [20] address the second.

## References

- [1] National Research Council, *Assessing the Reliability of Complex Models: Mathematical and Statistical Foundations of Verification, Validation, and Uncertainty Quantification*, The National Academies Press, 2012.
- [2] J. Pitt-Francis, P. Pathmanathan, M. Bernabeu, R. Bordas, J. Cooper, A. Fletcher, G. Mirams, P. Murray, J. Osborne, A. Walter, S. Chapman, A. Garny, I. van Leeuwen, P. Maini, B. Rodríguez, S. Waters, J. Whiteley, H. Byrne, D. Gavaghan, Chaste: A test-driven approach to software development for biological modelling, *Comput Phys Commun* 180 (12) (2009) 2452–2471.
- [3] J. Cooper, A. Corrias, D. Gavaghan, D. Noble, Considerations for the use of cellular electrophysiology models within cardiac tissue simulations, *Prog Biophys Mol Biol* 107 (1) (2011) 74–80. doi:10.1016/j.pbiomolbio.2011.06.002.
- [4] J. Cooper, G. Mirams, S. Niederer, High throughput functional curation of cellular models, *Prog Biophys Mol Biol* 107 (1) (2011) 11–20. doi:10.1016/j.pbiomolbio.2011.06.003.
- [5] A. Garny, D. Nickerson, J. Cooper, R. W. dos Santos, S. McKeever, P. Nielsen, P. Hunter, CellML and associated tools and techniques, *Phil Trans Roy Soc A* 366 (1878) (2008) 3017–3043. doi:10.1098/rsta.2008.0094.
- [6] M. Hucka, A. Finney, B. Bornstein, S. Keating, B. Shapiro, J. Matthews, B. Kovitz, M. Schilstra, A. Funahashi, J. Doyle, H. Kitano, *Evolving a lingua franca . . . : The Systems Biology Markup Language (SBML) project*, *Systems Biology* 1 (1) (2004) 41–53.
- [7] W3C RDF Working Group, *Resource Description Framework*, <http://www.w3.org/RDF/> (2004).
- [8] J. R. Terkildsen, S. Niederer, E. J. Crampin, P. Hunter, N. P. Smith, Using Physiome standards to couple cellular functions for rat cardiac excitation–contraction, *Exp Physiol* 93 (7) (2008) 919–929. doi:10.1113/expphysiol.2007.041871.
- [9] J. Cooper, S. McKeever, A model-driven approach to automatic conversion of physical units, *Softw Pract Exper* 38 (4) (2008) 337–359. doi:10.1002/spe.828.
- [10] VPH-FET Consortium, VPH-FET Research Roadmap, Ch. 4, <https://www.biomedtown.org/VPHFET/reception/> (Nov 2011).
- [11] D. Köhn, N. Le Novère, SED-ML — an XML format for the implementation of the MIASE guidelines, in: *Computational Methods in Systems Biology*, Vol. 5307 of LNCS, Springer Berlin / Heidelberg, 2008, pp. 176–190. doi:10.1007/978-3-540-88562-7\_15.
- [12] S. J. Dunn, P. Appleton, S. N. Nelson, I. Näthke, D. Gavaghan, J. Osborne, A two-dimensional model of the colonic crypt accounting for the role of the basement membrane and pericryptal fibroblast sheath, *PLoS Comp. Biol.* 8 (2012) e1002515. doi:10.1371/journal.pcbi.1002515.
- [13] G. T. Eisenhoffer, P. D. Loftus, M. Yoshigi, H. Otsuna, C.-B. Chien, P. A. Morcos, J. Rosenblatt, Crowding induces live cell extrusion to maintain homeostatic cell numbers in epithelia, *Nature* 484 (2012) 546 – 549.
- [14] G. De Matteis, A. Graudenzi, M. Antonietti, A review of spatial computational models for multi-cellular systems, with regard to intestinal crypts and colorectal cancer development, *Journal of mathematical biology* (2012) 1–54.
- [15] J. M. Osborne, A. Walter, S. K. Kershaw, G. R. Mirams, A. G. Fletcher, P. Pathmanathan, D. Gavaghan, O. E. Jensen, P. K. Maini, H. M. Byrne, A hybrid approach to multi-scale modelling of cancer, *Phil. Trans.* (2010) 5013–5028.
- [16] I. M. M. Van Leeuwen, G. R. Mirams, A. Walter, A. Fletcher, P. Murray, J. Osborne, S. Varma, S. J. Young, J. Cooper, J. Pitt-Francis, L. Momtahan, P. Pathmanathan, J. P. Whiteley, S. J. Chapman, D. J. Gavaghan, O. E. Jensen, J. R. King, P. K. Maini, S. L. Waters, H. M. Byrne, An integrative computational model for intestinal tissue renewal, *Cell Prolif.* 42 (2009) 617 – 636.
- [17] G. Mirams, A. Fletcher, P. Maini, H. Byrne, A theoretical investigation of the effect of proliferation and adhesion on monoclonal conversion in the colonic crypt, *Journal of Theoretical Biology* 312 (7) (2012) 143–156. doi:10.1016/j.jtbi.2012.08.002.
- [18] W. Wong, N. Mandir, R. Goodlad, B. Wong, S. Garcia, S. Lam, N. Wright, Histogenesis of human colorectal adenomas and hyperplastic polyps: the role of cell proliferation and crypt fission, *Gut* 50 (2) (2002) 212–217.
- [19] The HDF Group, *Hierarchical data format version 5*, <http://www.hdfgroup.org/HDF5> (2000).
- [20] G. Wilson, Software carpentry: Getting scientists to write better code by making them more productive, *Comput Sci Eng* 8 (6) (2006) 66–69.  
URL <http://www.software-carpentry.org/>