Syst Synth Biol (2012) 6:35–49
DOI 10.1007/s11693-012-9095-x

RESEARCH ARTICLE

On an infrastructure to support sharing and aggregating pre- and post-publication systems biology research data

Mark Slaymaker · James Osborne · Andrew Simpson · David Gavaghan

Received: 12 March 2012 / Revised: 14 June 2012 / Accepted: 23 July 2012 / Published online: 3 August 2012

Abstract The move towards in silico experimentation has resulted in the use of computational models, in addition to traditional experimental models, to generate the raw data that is analysed and published as research findings. This change requires new methods to be introduced to facilitate independent validation of the underlying models and the reported results. The promotion of co-operative research has the potential to help to both validate results and explore wider problem areas. In this paper we leverage and extend two existing software frameworks to develop an infrastructure that has the potential to both promote the sharing of data between researchers pre-publication and enable access to the data for interested parties post-publication. The pre-publication sharing of data would enable larger problem spaces to be explored by distributed research groups; enabling access to the data post-publication would allow reviewers and the wider community to independently verify the published results which would, in the longer term, help to increase confidence in published results. The framework is used to perform reproducible and numerically validated individual-based computational experiments into the onset of colorectal cancer. Existing results are verified and new insights into the top-down versus bottom-up hypothesis of colorectal crypt invasion are given.

Keywords Systems biology · Data management · Colorectal cancer

M. Slaymaker (✉) · J. Osborne · A. Simpson · D. Gavaghan
Department of Computer Science, University of Oxford, Wolfson Building, Parks Road, Oxford OX1 3QD, UK
e-mail: mark.slaymaker@cs.ox.ac.uk

J. Osborne
e-mail: james.osborne@cs.ox.ac.uk

A. Simpson
e-mail: andrew.simpson@cs.ox.ac.uk

D. Gavaghan
e-mail: david.gavaghan@cs.ox.ac.uk

Introduction

In silico experimentation, whereby computational modelling and simulation are used to understand and interpret biological data, is gaining increased use in systems biology. As these simulations become increasingly complex and often build on existing published work, the need for methods which enable the validation and reproduction of results is crucial. The central dogma of systems biology considers a system as a whole entity rather than taking a more traditional reductionist view and investigating individual components in isolation (Kitano 2002). Such an approach requires more sophisticated models and for these models to be realistic and useful; in silico results will directly influence experimental work and vice versa—the *virtuous cycle* (Di Ventura et al. 2006)—further increasing the need for accurate and reproducible simulations.

When publishing experimental work all components and processes used are recorded and presented allowing the results to be reproduced and extended. If we hope to be able to make progress with simulations then we need a similarly rigorous approach. Currently such standards are not applied to computational simulations and therefore verifying and extending results is a laborious process.

The validation of a published mathematical model is also a well-established process: the mathematical analysis published in the paper can be verified by the reader; furthermore, many mathematical models can be rendered in some form of

mark-up language such as CellML[1] or SBML[2] to ensure the consistency and reproducibility of the model and results. For example, methods for the validation of computational simulations are less developed; however, methods for validation and verification of computational models of processes operating on single scales such as biomechanics have been proposed in Anderson et al. (2007). These approaches are not currently sufficient when dealing with complex multi-scale multiphysics computational models as it is not usually possible to render such models using existing mark up languages, and the code used to produce simulation results is more complex than that of phenomena occurring at a single scale. In addition to the soundness of the simulation software being used, there also needs to be confidence in the analysis of any results and confidence that appropriate ranges of starting parameters have been considered.

In April 2001 an *e-Science Programme* was launched in the UK, with the intention of utilising advanced computing techniques to provide researchers with access to widely distributed data and computational resources to facilitate 'bigger' and 'better' research. A number of projects worked on various aspects of these high-level goals. Mechanisms for providing interoperability of web services by using mediator-based composition were considered in Radetzki and Cremers (2004), with the mediators' capabilities being defined using the Ontology Web Language (OWL)[3]. Projects such as eDiaMoND (Brady et al. 2003) provided the groundwork upon which the GIMI (Generic Infrastructure for Medical Informatics) project (Simpson et al. 2010a) built, by developing a middleware framework designed to support the secure federation of distributed, heterogeneous data sources. (This work is leveraged in this paper.) As a further example, the provision of providence data in bioinformatic experiments was considered in the work of the ^{my}Grid (Greenwood et al. 2003) project team. In addition to providence information it is necessary for wider community involvement in the validation of published data especially data generated by in silico experimentation. If the long-term goals of the programme and subsequent initiatives are achieved, we will arrive at a scenario in which large volumes of high quality research data may be shared by researchers, allowing them to investigation of a wider range of problems.

Facilitating the sharing of pre-publication research data between collaborating research groups could enable a wider spectrum of experimental data to be combined for analysis. One example of how this could be achieved is the partitioning of the parameter space of interest between several research groups: each group might then run simulations using their portion of the parameter space and record the results. Using a middleware framework that facilitates the aggregation of distributed data could allow these groups to be distributed across a wide geographic area, with the middleware allowing the combination of the subsets of results to facilitate analysis. In addition to the analysis of data generated in a co-operative way, further analysis could be performed on larger sets of general simulations, generated by many different research groups, to help identify areas of potential interest which warrant further study.

The ability to make versions of the software used in generating the results (as well as the actual experimental results and raw data) available to the wider community would permit interested parties to verify the correctness of the results claimed in publications. In addition to assisting validation it would be possible to perform further statistical or other analysis of the results.

Of course, the desire to have access to algorithms and results is nothing new: for example, in Ignizio (1971) the author proposes a standard method for the reporting of results relating to algorithm execution to permit comparison. This was followed up by Ignizio (1973) in which the author suggested that a more rigorous approach to the assessment of claims in publications relating to algorithms is needed, including the need to submit copies of code. Although the ideas presented were quite onerous and technologies have moved on, many of the general points raised are still valid today—almost 40 years later. Providing other interested groups with enough information to re-run the experiments would help in achieving the objective of verifying that published results are repeatable.

The importance of the reproducibility of results when conducting experiments based upon mathematical software is considered in Crowder et al. (1978), a section of which is consists of a discussion on the meaning of the reproducibility of results. The authors concede that there are inherent problems in expecting extreme accuracy when attempting to reproduce results. A number of these problems relate to the variability and continual change of the infrastructure used, in particular hardware, software, operating systems and programming languages. The consideration of these issues lead the authors to suggest the following description as a reasonable meaning of reproducibility:

> Thus, when requiring reproducibility, we do not mean a precise replication of results. Rather, a set of results that agree with the original to within a tolerance that may reasonably be attributed to changes in technology.

The authors also accept that it is difficult within the confines of a published article to provide sufficient information to achieve this goal, and suggest a method of reporting and supporting documents that would help with achieving the aim of reproducibility of Crowder et al. (1978).

[1] http://www.cellml.org/.

[2] http://sbml.org/.

[3] http://www.w3.org/TR/owl-semantics/.

In this paper we describe a method of providing access to data analysed to generate results and a way of re-executing the original application that was used to generate the data. This allows analysis on larger data sets, as well as allowing confirmatory analysis to be carried out to verify published results. The executable used for generating the data processed also processes the simulation results. The processing populates a database with information relating to the exact version of the software used, the version of support libraries, the hardware the simulation was run on, the operating system being used, and the parameters used to invoke the simulation. The overall purpose of the proposed framework is to provide a mechanism that supports our key aims, which can be summarised as:

- reproducibility of results,
- validation of results,
- facilitating the extension and re-analysis of results by enabling data aggregation, and
- permitting the generation of results by running new simulations.

We feel that by meeting these aims the approach detailed in this paper is a step towards the goals discussed in Ignizio (1971, 1973) and Crowder et al. (1978).

The focus of this paper is Chaste (Cancer, Heart and Soft Tissue Environment)[4], an open source library for the simulation of problems in Computational Biology. The main applications of Chaste are cardiac electrophysiology and discrete modelling of populations of cells which are considered in Pitt-Francis et al. (2009). Being open source, simulations created using Chaste can be run by any interested party to validate and verify results; however there is currently no process for the raw results themselves to be distributed. The framework *sif* (for service-oriented interoperability framework) (Simpson et al. 2010a), originally developed in the aforementioned GIMI project, is a mechanism for the controlled sharing of data. It is via sif that we are able to give community access to the raw data that the published results are based on.

The structure of the remainder of the paper is as follows. A description of the infrastructure is given in section "Supporting infrastructure", which includes details of Chaste, and the sif middleware. In section "An example application: a model of colorectal cancer" we present an example of an in silico experiment using Chaste which is used to illustrate how the sif middleware could be used to aggregate results from multiple simulations, and allow the wider community access to the results. In section "Realising the 'vision'" we discuss how we intend to address our longer term 'vision'. Finally, section "Conclusions" provides concluding comments and identifies possible directions for future work.

Supporting infrastructure

In this section we provide brief overviews of both the computational modeling framework, Chaste, and a secure middleware framework which provides facilities allowing sharing of data, files and the remote execution of applications: sif.

Chaste

Chaste has been designed to act as a high-quality multi-purpose library supporting computational simulations for a wide range of biological problems, as considered in Pitt-Francis et al. (2008). In this context, 'high-quality' means that the software is extensible, robust, fast, accurate and maintainable, and uses state-of-the-art numerical techniques. It is also open-source, and so can be adapted, extended and validated by other developers as appropriate. Chaste has been developed by a multidisciplinary team including mathematicians and software engineers using agile development techniques. This has helped ensure that the code is well-structured as a piece of software, while at the same time is practical and useful as a computational modelling tool. While it is a generic, extensible library, to date attention has focused on the fields of cardiac electrophysiology and modelling of colorectal cancer (Pitt-Francis et al. 2009).

Chaste has a modular structure and is composed of a core set of libraries which provide the central functionality. Application-specific components are able to link to this core code to enable simulations of differing biological systems to be undertaken using multiple modelling techniques (discrete and continuous, both deterministic and stochastic). The interested reader is referred to Pitt-Francis et al. (2009) for details of this structure.

The work presented in this paper builds upon the cell-based (discrete modelling) component of Chaste (hereafter referred to as Cell-Based Chaste), but it could be applied to any simulation software. As discussed, there are two main applications of Chaste; for further details of the cardiac side of Chaste, where electrical propagation in the heart is modelled using a reaction diffusion system (known as the Bidomain Equations) coupled to cell level models for ion flow in individual cells, see Pitt-Francis et al. (2009) and Pathmanathan et al. (2010).

The cell-based side of Chaste is used to run multi-cell multi-scale simulations of biological systems. Simulations are based upon the principle that the cell is the natural structural unit in biology and the framework consists of

[4] http://www.cs.ox.ac.uk/chaste.

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 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 ODEs. 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 concentrations, with cells functioning as sinks and sources. This modular approach enables multiple possible models to be considered at each scale. The framework can be used to create simulations of many biological systems. Pitt-Francis et al. (2009). One particular application is the onset of cancer in the colorectal crypt, illustrated in Osborne et al. (2010), and which is discussed in detail in section "An example application: a model of colorectal cancer".

sif

The sif middleware framework was initially developed to support a variety of medical research applications. The nature of the domain led to a requirement for a robust mechanism to enforce access control policies; subsequently, sif has developed into a generic system that can be utilised in a wide variety of application areas.

sif has been utilised to provide secure distributed access to resources to enable federation of data in several contexts including health—considered in Slaymaker et al. (2008b)—and systems biology—detailed in Simpson et al. (2010b). In Slaymaker et al. (2008a), a key tenet of sif is described: that the resource owner maintains control over who has access to their data. This is achieved by the use of fine-grained access control policies, with each policy relating to an individual resource and the policy being defined by the owner of the resource being protected. In addition to the individual policies relating to single sources, the middleware has a central policy which itself allows the development of policies controlling who can initiate interactions with the system.

The sif middleware also provides a way to add additional functionality via a 'plug-in' mechanism, which presents a standard interface to resources accessed by one of the three supported plug-in types: data plug-ins, file plug-ins, and algorithm plug-ins. Data plug-ins provide access to data resources via a subset of SQL, with the plug-in writer being responsible for any necessary translation between the data sources's native interaction and SQL. A file plug-in presents a resource to be exposed as if it were a file system. Finally, an algorithm plug-in provides a facility for encapsulating and remotely invoking an algorithm. This is achieved by providing a standard way of defining the inputs required and outputs generated by the algorithm. The ability for any user to perform an action requires it to have been permitted by the access control mechanism.

The development of a number of applications that utilise the sif middleware has been facilitated by the provision of a Java application programming interface (API). The API abstracts away from the more complicated concerns relating to web services and certificate handling, allowing developers to concentrate on the functionality of their application. In addition to the API, other aspects of functionality include a generic data plug-in, which, with minimal configuration, can be used to access JDBC (Java Database Connectivity) data sources.

Asynchronous execution

The default mode of communication between a client application and sif is synchronous: the client application makes a request and waits until it receives a response. The response can take one of several forms, with the result expected indicating: that the request has completed successfully; a general failure in the processing of the request; or that a time-out has occurred.

The standard time-out period for web services is typically in the order of several minutes, which is suitable for jobs with a relatively short execution time. It is, however, inappropriate for jobs which last for many minutes or hours. Although the time-out period could be extended, it is not a practicable solution to have a time-out period of several days—as it would require the connection between the client and server to be maintained throughout effectively blocking the client. An additional problem associated with a very long time-out period is that, in the case of a job that takes longer than the time-out period, the client will wait until the time-out and still not receive any useful results.

Other scenarios in which a synchronous connection is not always desirable include cases where the quality of the connection may be such that a job can be successfully submitted but not maintained long enough to receive the results. Alternatively, it can sometimes be useful to submit a number of jobs for which the results are not needed immediately but would be useful to retrieve later or from a different location.

To provide a facility for these scenarios and to overcome the limitation on the maximum time a job can take to execute, an asynchronous mode of executing algorithm plug-ins has been integrated into the middleware. The asynchronous mechanism permits any previously defined

algorithm plug-in to be executed in an asynchronous fashion without any need to rewrite the plug-in. This extension in functionality was implemented to enable aspects of the work described in this paper.

The contributions of Brambilla et al. (2004) and Zdun et al. (2004) describe a number of patterns of asynchronous web service behaviour, with the *polling pattern* with possible email notification being the pattern utilised in the following for managing asynchronous plug-in calls. This pattern was chosen because most of the other patterns require the client to publish services which can be used for receive notification of the jobs completion via a callback mechanism. Typical use cases involve deployment of clients within an institution which have policies in place that do not permit clients to publish services (usually due to firewall configurations placing restrictions on which open ports are permitted).

Collaboration

The vision that drives our work is depicted in Fig. 1. This illustrates how different research groups may co-operate to generate a published paper and then permit reviewers and the wider community to have access to data for validation purposes. The left-hand side of the figure shows two (of possibly many) research groups. The upper group, via a Chaste application, use a plug-in within the sif middleware to execute a cell-based Chaste simulation and collect the data. The lower group execute the cell-based Chaste simulation manually and then process the results into a database. All the co-operating groups, indicated by the lighter dotted elements in the middle of the left-hand side, use the same query application (with their own credentials) to access whichever portions of the aggregated data they have the authority to view.

The right-hand side shows reviewers and members of the wider community accessing the data via a community query application. This application would be made up of pre-defined queries that would allow access to the data which underpinned the published results, allowing interested parties to check the conclusions drawn in a publication. The community query application would be supplied with a guest certificate which would permit access only to the data sufficient to allow validation to occur. Although the application would run only a set number of pre-defined queries, it should allow some customisation via a range of parameters.

In addition, the ability to retrieve the results of executions may allow the wider community to validate any analysis performed. This could be achieved by making available a community certificate and a simple query application, which would allow interested parties to access a publicly available version of data which was used in generating published results. The process of providing access to the raw data will help to strengthen confidence in published results as well as allow the community access to a wider pool of results that can be compared to ascertain consistency or suggest new lines of enquiry.

The ability to retrieve the starting parameters of simulations along with the version of the code used to generate the results would also aid the validation process allowing other groups to re-run experiments, as well as to perform their own analysis to validate findings and compare the results with other methods.

This illustrative overview only shows the use of a Chaste-based application to generate the data being shared; however, there are no inherent restrictions on how the data being shared is created or, indeed, on the number of different data sources being combined. This is to enable many different groups to generate data appropriate to their own needs and then share it as they see fit with other groups within the community.

An example application: a model of colorectal cancer

As discussed previously, the current method of in silico experimentation involves individuals (or groups) running simulations. In the best case scenario, the code to run simulations is stored; however, the details of the processes applied to the simulations both in terms of parameter values used and subsequent analysis is not. We have developed a process which uses the Chaste software to generate simulation results, and the sif middleware to execute the simulations, populate data sources for future querying and return requested results to allow analysis of the data generated. In this section we present this process—from the perspective of an end user—by using the example of simulations of the colorectal crypt.

The crypt

The lining of the intestine is comprised of a large number of test-tube shaped crypts. These crypts are spaced regularly throughout the intestine and provide a layer of epithelial cells which form the lining of the intestine and protect the underlying tissue, with the intestinal epithelium being renewed every few days. This renewal is the result of coordinated proliferation, migration, differentiation and apoptosis within the crypt. It is understood that stem cells, which reside at the base of each crypt, divide and produce transit cells, which, in turn, divide and migrate up the crypt. Once cells reach a certain height up the crypt they differentiate and no longer divide; this is due to the levels of signalling molecules which vary along the crypt influencing the cell through the Wnt cascade (Van Leeuwen et al. 2009).

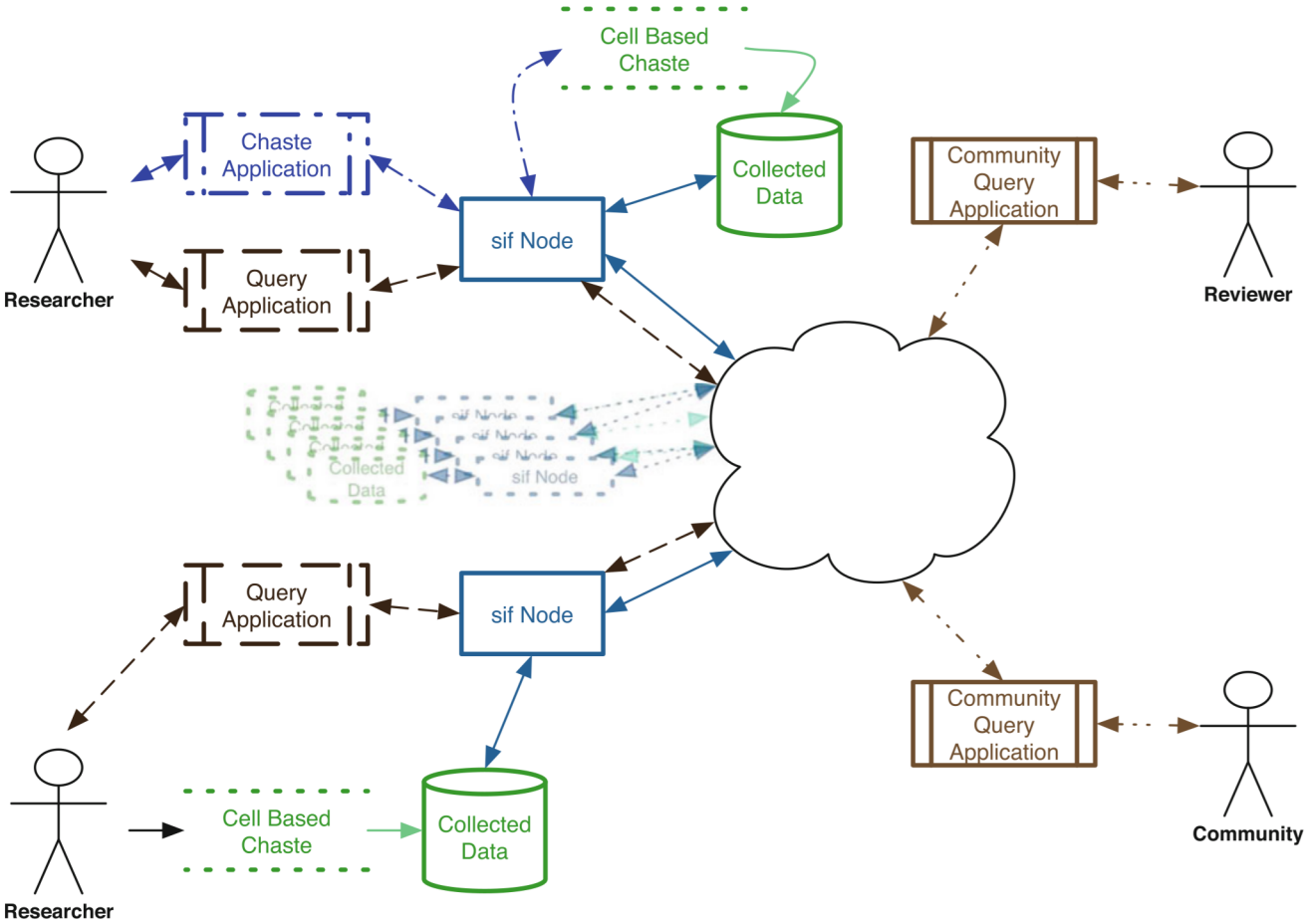

Fig. 1 Generating and federating results from multiple sources

Colorectal cancer (CRC) is the third most common malignancy in the United Kingdom[5] and originates from the epithelium that lines the colonic crypt. The natural balance between proliferation, migration and apoptosis is destroyed via a series of mutations that cause cells: (1) to ignore environmental cues that normally halt cell proliferation; and (2) to adhere to neighbouring cells more strongly. Such mutations can destabilise the healthy crypt structure, causing the crypt to buckle and, eventually, to form a tumour.

Over the last 30 years many mathematical models have been proposed to represent the colonic crypt. Early models of the crypt took the form of cellular automata describing the behaviours of individual cells which are restricted to lie on a uniform rectangular lattice (Loeffler et al. 1986). These models have since been extended, and the restriction that cells lie on a fixed lattice has been removed by using an off-lattice tessellation based cell centre model (Meineke et al. 2001). Further extensions have concentrated on: adding more realistic details for cell proliferation; considering the influence of external signalling factors (Van Leeuwen et al. 2009); and including mutations in the model to be able to consider the onset of CRC (Osborne et al. 2010). These models have made progress by considering the crypt to be an unwrapped cylinder, essentially reducing the problem to two spatial dimensions. However, recent progress has been made in representing the crypt in three dimensions by considering a fixed and realistically-shaped imposed geometry (Buske et al. 2011).

CRC arises when the detailed balance (between proliferation, migration and apoptosis) present in the healthy crypt is lost through the acquisition of mutations in individual cells, which affect the properties of individual cells and their progeny. The site and type of these mutations is critical when trying to understand the development of CRC. Osborne et al. (2010) show that the distance of the mutation from the crypt base plays a key role in whether the progeny of the mutated cell can take over the crypt. The remainder of this section outlines the computational model for the onset of CRC presented in Osborne et al. (2010), before giving details of the coupling of this model to the sif framework. The section concludes by utilising this new computational framework to validate and extend results from a previously published paper, Osborne et al. (2010).

[5] http://www.cancerresearchuk.org/.

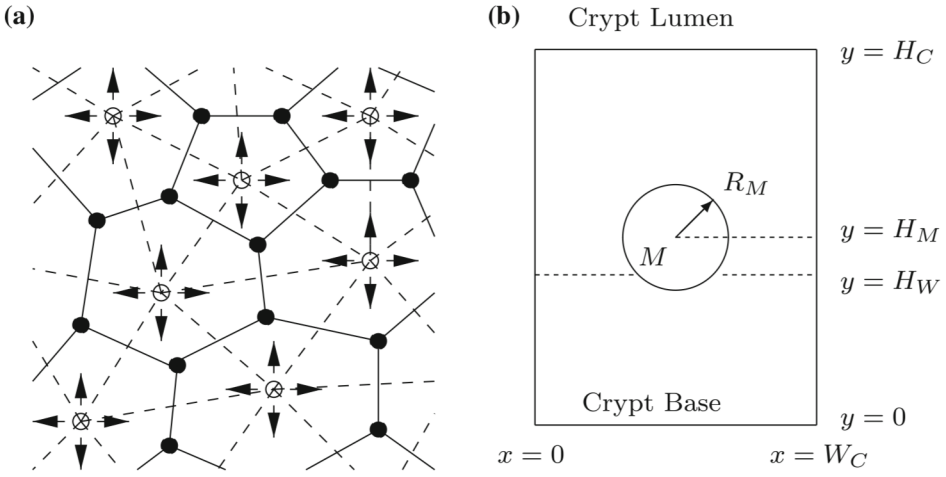

Fig. 2 **a** Schematic representations of the cell-level model (Cell-centre model) the size and shape of the cells (bounded by *solid lines* and *filled circles*) are determined from a Voronoi tessellation of the cell centres (*empty circles*). **b** Schematic representation of the crypt including simulation variables. Initially cells in the region M are defined as mutant. Figures reproduced with permission from Osborne et al. (2010)

Model of the crypt

Following Van Leeuwen et al. (2009) and Osborne et al. (2010) we consider the crypt as a two-dimensional cylindrical surface of height H_C cell diameters and circumference W_C cell diameters (shown in Fig. 2b), where we estimate that a cell diameter is approximately 20 μm (Alberts et al. 2002). Circumferential distance is represented by $x \in (0, W_C)$ and the distance from the crypt base is denoted by $y \in (0, H_C)$. Periodic boundary conditions are imposed on $x = 0$ and $x = W_C$. Cells are removed, or sloughed off, when they reach the lumen ($y = H_C$) and cells cannot pass through the base of the crypt ($y = 0$). In addition, there is a spatial gradient of Wnt (a signalling molecule which affects cell proliferation) along the crypt axis, so that Wnt levels are high near $y = 0$ and low near $y = H_C$.

Individual cells are modelled using a lattice-free, cell-centre model (see Fig. 2a) described by Van Leeuwen et al. (2009), Osborne et al. (2010). Each cell is treated as a discrete entity represented by its centre. Adjacent cell centres are connected by linear springs where neighbouring cells are determined by a Delaunay triangulation. Moreover, cell shapes are determined by a Voronoi tessellation.

The equations of motion are

$$\mu_i \frac{\mathrm{d}\mathbf{r}_i}{\mathrm{dt}} = \sum_{j \in S_i} k_{ij}\left(|\mathbf{r}_i - \mathbf{r}_j| - s_{ij}(t)\right) \frac{(\mathbf{r}_j - \mathbf{r}_i)}{|\mathbf{r}_j - \mathbf{r}_i|}, \qquad (1)$$

for $i = 1, \ldots, n$, where $\mathbf{r}_i$ is the position of cell-centre i, n is the total number of cells, $s_{ij}(t)$ is the natural length of the spring connecting cell-centres i and j, k_{ij} is the spring constant of the spring connecting cell-centres i and j, S_i is the set of neighbouring cells of cell i, μ_i is the drag coefficient, which depends on cell i's type (and represents the effects of cell stromal adhesion), and t is time.

These equations are derived by balancing viscous drag with cell-cell interaction forces associated with the compression and extension of the springs and neglecting inertial effects; more details can be found in Van Leeuwen et al. (2009).

When cell i divides, its daughter, cell j, is placed (in a randomly chosen direction) 0.05 cell diameters from cell i, and cell i is moved 0.05 cell diameters in the other direction. To model cell growth during the M-phase of the cell-cycle, the rest length of the spring connecting parent and daughter cell, $s_{ij}(t)$, increases linearly from 0.1 to 1 cell diameters during the last hour of the cell cycle (the first phase to occur after a new cell is created), and we set $s_{ij}(t) = 1$ cell diameter for all other connections.

We suppose that the crypt contains healthy and mutant cells which are assumed to differ in their rates of cell proliferation and their cell–cell and cell–substrate adhesion properties. Following Osborne et al. (2010), we assume that healthy cells proliferate in the region of the crypt, $0 \leq y \leq H_W$, where Wnt levels exceed a prescribed threshold level. By contrast, mutant cells proliferate throughout the crypt, irrespective of the local Wnt concentration. The mutant cells are also assumed to exhibit increased stromal adhesion and stronger cell-cell adhesion modelled through the μ_i parameter.

The cell cycle of both normal and mutant cells is assumed to vary stochastically. We set the total S, G_2 and M phase duration as 10 h. The duration of the G_1 phase is sampled from a normal distribution, with mean of T_{g_1} hours and standard deviation of 1 h. In Osborne et al. (2010) the default value of $T_{g_1} = 6\,\mathrm{h}$ is used to make the total cell cycle duration have a mean of 16 h.

In order to study the effect of mutant cells on the crypt we perform the following experiment: (1) given a specified random seed, τ, we run the crypt model described above to homeostasis (until $t = T_S$); (2) once in homeostasis a patch of mutant cells is introduced [see Fig. 2b (any cell with a centre within R_M cell diameters of the point (W_C/2, H_M) is set to be mutant)]; and (3) the simulation is run until a specified end time ($t = T_S + T$), with the number of each type of cell and the height of the base of mutations (defined

```
<Chaste>
  <CryptSimulation2dMutantTrack>
    <RandomSeed>1</RandomSeed>
    <TimeToSteadyState>100</TimeToSteadyState>
    ...
    ...
  </CryptSimulation2dMutantTrack>
  <MeshBasedCellPopulationWithGhostNodes-2>
    <GhostSpringStiffness>15</GhostSpringStiffness>
    ...
    ...
  </MeshBasedCellPopulationWithGhostNodes-2>
  <CellCycleModels>
    ...
    ...
  </CellCycleModels>
  <Forces>
    ...
    ...
  </Forces>
  <CellKillers>
    ...
    ...
  </CellKillers>
  <CellPopulationBoundaryConditions>
    ...
    ...
  </CellPopulationBoundaryConditions>
</Chaste>
```

Fig. 3 Outline structure of the parameters file produced by Chaste

as the y coordinate of the lowest mutant cell) over time being tracked.

Implementation

In order to investigate the effect of mutant cells on the crypt, an executable was created using Chaste. This executable enables a subset of simulations possible within Chaste to be run as simply as possible. All parameters used to set up the simulation in Chaste are output to a parameters file. An example of such a parameters file is given in Fig. 3, which includes all of the parameters which are set at run time and also the default parameters which are fixed for this example application. This is used to facilitate querying of the data and to specify all the parameters used in the simulation. This format is specific to Cell-Based Chaste; another format could be used by an alternative piece of simulation software so long as sufficient information to fully specify the simulation was included.

The executable runs simulations of the crypt model described in section "Model of the crypt". The model parameters specified in Table 1 can be varied. The remaining model setup and parameters used are described in section "Model of the crypt" and in Osborne et al. (2010). Figure 3 illustrates the structure of the XML file that records all the parameters used by the simulation which includes both the parameters varied in the simulation as well as those that are fixed. The capture of these details greatly enhances the possibility of being able to reproduce the results of a given simulation. The main outputs of these simulations are data files containing the position (along with information about the connectivity) and type of cells over time, allowing results to be visualised (an example of the visualisation is given in Fig. 4). In addition to the visualisation, the number of cells (of each type—healthy and mutant) in the crypt and the base of the mutant 'blob' over time are tracked in each simulation and stored in separate text files.

An algorithm plug-in was also developed to permit the remote and asynchronous execution of the executable. The plug-in also recorded the parameters used (both passed to the executable and also those used to set up the executable in Chaste) and the location of any results files. The recording of parameters and version information facilitates the re-running of any simulation to verify that the results are consistent.

An example computational application

A prototype application was created to interface with the executable via an algorithm plug-in. This application, illustrated in Fig. 4, has fixed input panes that correspond to the various tasks that are required which include: parameter entry for a single execution of the plug-in; ranged parameter entry to allow a number of jobs to be executed; a list of currently active jobs; a pane to allow queries to be written (which interacts with a data plug-in); and a results pane which allows the data to be retrieved and viewed. The final pane allows the user to recover the subset of data that they are interested in for a given simulation run—which is achieved via interaction with a file plug-in.

In the following sections we present four examples where the framework presented is used to:

- reproduce published results;
- validate these results;
- perform a re-analysis of these results; and
- generate new results by running new simulations.

Reproducing results from Osborne et al. (2010)

As argued previously, one of the key features of any simulation curation system is the ability to reproduce simulations results; moreover, this is of fundamental importance as it allows in silico results to be considered on the same level as experimental results. To illustrate the value of an application such as that of the previous section in this regard we use the executable to reproduce some of the results of Osborne et al. (2010).

By using the default parameters of section "Implementation" and performing a sweep over the following variables, we are able to reproduce Fig. 8a, d from Osborne et al. (2010):

Table 1 Table of parameters which can be specified in a simulation

Description	Parameter	Default value	Constraints
Crypt length	H_C	20	$H_C > 0$
Crypt circumference	W_C	10	$W_C > 0$
End time	T	50 h	$T > 0$
Random seed	τ	1	None
Wnt proliferating threshold	T_{g_1}	2/3	$0 \leq H_W \leq 1$
Cell cycle G_1 duration	T_{g1}	6 h	$T_{g_1} \geq 0$
Initial height of mutant region	H_M	2	$H_M \geq 0$
Initial radius of mutant region	R_M	2	$R_M > 0$
Mutant viscosity multiplier	μ_M/μ_N	1	$\mu_M/\mu_N > 0$
Time till steady state	T_S	100 h	$T_S \geq 0$
Time step	dt	1/120 h	$dt \geq 0$

- $H_M \in \{4{:}4{:}16\}$;
- $\mu_M/\mu_N \in \{1{:}1{:}20\}$; and
- $\tau \in \{1{:}1{:}100\}$.

Figure 5 shows a comparison of the results published in Osborne et al. (2010) and the results produced by our framework. As in Osborne et al. (2010), results are averaged from 100 simulations for each viscosity ratio, μ_M/μ_N, and $B(t)$ denotes the height of the lowest mutant cell in the crypt. It is clear from comparisons between Fig. 5a, b and Fig. 5c, d, respectively, that the results reproduced by the framework are qualitatively the same as those published in Osborne et al. (2010).

To generate these curves for comparison, results from multiple (stochastic) simulations are post-processed and averaged in order to generate global 'tissue-level' properties. There may be small differences introduced (for example, the $t = 50$ curve in Fig. 5b is slightly higher than the corresponding curve in Fig. 5a) because the parameters used to generate the results used in the earlier publication were not rigorously recorded. However, it is the shape of these curves—which show tissue-level behaviour—rather than specific numerical values that are important when making comparisons. We could also compare other tissue-level properties, such as velocity profiles or cell compression within the crypt [see Osborne et al. (2010)]; however, due to the averaged nature of these properties, an exact quantitative agreement between results will still not be possible. It is also worth remembering the comments of Crowder et al. (1978), which accept that there may be differences in results which can be attributed to changes in technology. It is hoped that by using the framework

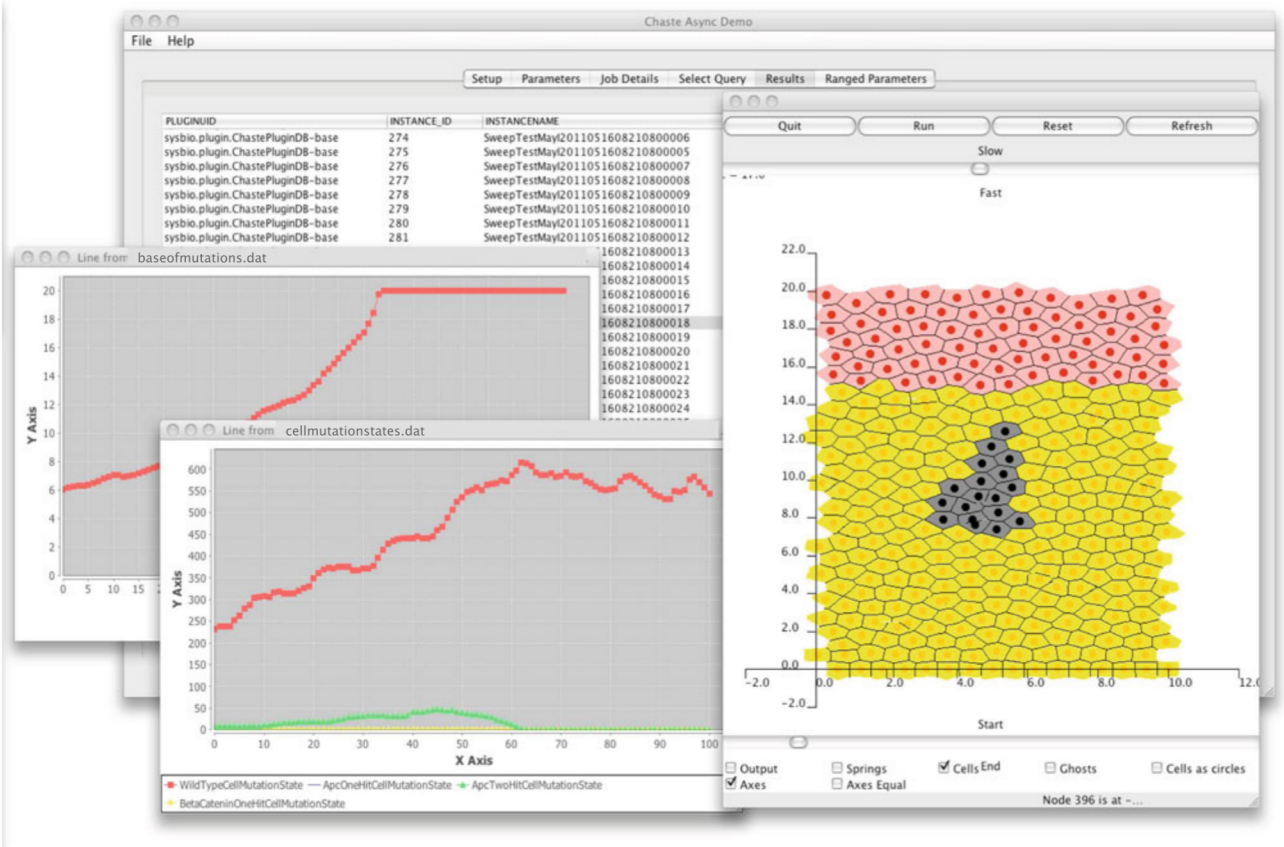

Fig. 4 Sample application retrieving aggregated results

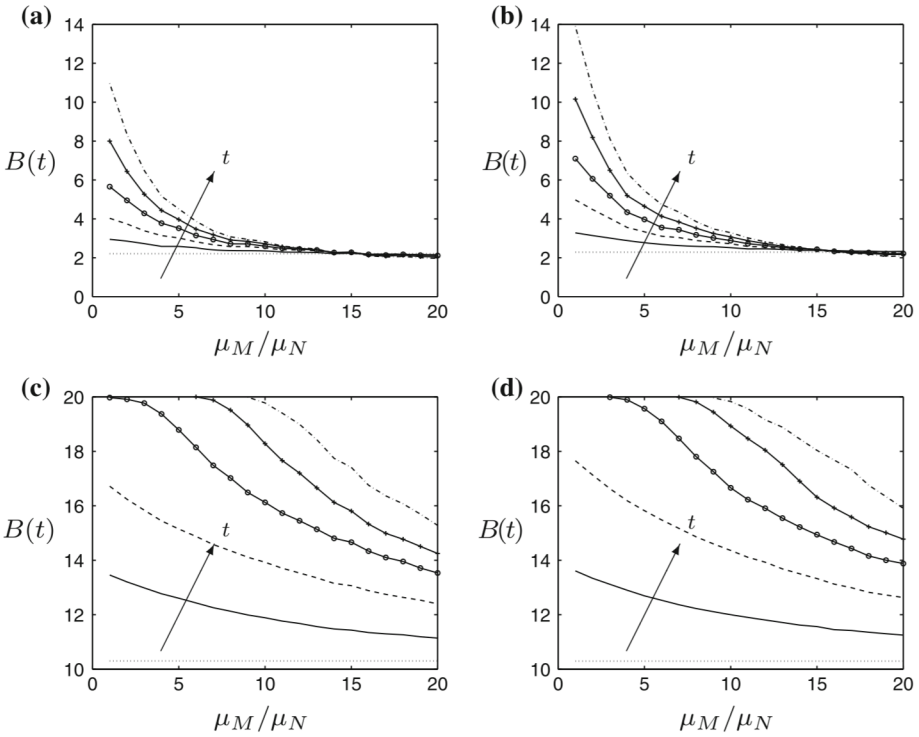

Fig. 5 Plots showing how the vertical height of the centre of the lowest cell in the mutant patch ($B(t)$) changes over time as the initial vertical position of the mutant patch and the corresponding drag ratio vary. The initial patch is a circle of radius 2, centred at $(x, y) = (W_c/2, H_M)$ and results are plotted at times $t = 0$, 10, 20, 30, 40, 50 h. **a**, **c** The results from Osborne et al. (2010) and **b**, **d** The results using the sif framework. Key: $H_M = 4$ for (**a**, **b**); $H_M = 12$ for (**c**, **d**)

presented here that this type of systematic difference can be greatly reduced; moreover, by recording the versions of software used, along with system information and seeds used by the chosen random number generators, an exact reproduction of results would be possible.

Verifying results

In addition to re-running experiments, the framework can be used to validate the numerical methods used to create the simulations. This is the second of the key features of simulation curation proposed in the introduction.

The influence of varying the timestep, dt, on experimental output is investigated in Fig. 6a. As before, results are averaged from 100 simulations for each viscosity ratio; however, here we only plot the base of the mutant population at time $t = 20$ to be able to compare the results clearly. If the timestep is doubled (to $dt = 1/60$), then the results are drastically different—as seen by the dotted line in Fig. 6a. This is because, with a larger timestep, the numerical method used to update the positions of the cells (Eq. 1), the forward Euler method, becomes unstable and cells move in a haphazard manner. As seen by the solid and dashed lines, when the timestep is halved from the default value (to $dt = 1/240$) then the solution is similar showing the numerical method has converged with the default timestep of $dt = 1/120$ h. All of the other results presented in Fig. 5 are affected in a similar manner (with the results being omitted for reasons of brevity).

In order to see if the simulation has reached a homeostatic steady state before we introduce mutant cells, we can investigate the influence of the time until steady state parameter, T_S. To this end, Fig. 6b compares the results from Fig. 5b, d ($T_S = 100$, in black) with simulations where $T_S = 0$ (in red) and $T_S = 150$ (in blue). If the simulation is not at a steady state before the mutant cells are introduced, then the results are different. With $T_S = 0$, initially the mutant cells are slightly higher than when $T_S > 0$, due to the regular lattice structure of the initial conditions (results not shown). However, at later times the height of mutant cells is lower for $T_S = 0$; this is because when the system is in homeostasis the cells are compressed [as shown in Osborne et al. (2010)], and, therefore, for $T_S > 0$, there will more pressure from the base of the crypt causing the upward migration of cells; for $T_S = 0$ we first need to attain this compressed state causing cells to be lower in the crypt. This is most marked for the case of $H_M = 12$ (as shown in Fig. 6b) as there are more cells below the mutant blob in the crypt. Most importantly, the curves for $T_S = 100$ and $T_S = 150$ are practically indistinguishable—showing that we are in a steady state when introducing mutant cells for the original simulations. All the other results presented in Fig. 5 are affected in a similar manner (with the results being omitted for reasons of brevity).

Extending results using existing data

In addition to being able to validate and verify the simulation results—which the framework allows to be done in an automated manner—we may also use the framework to extend the results initially presented. Moreover, we are

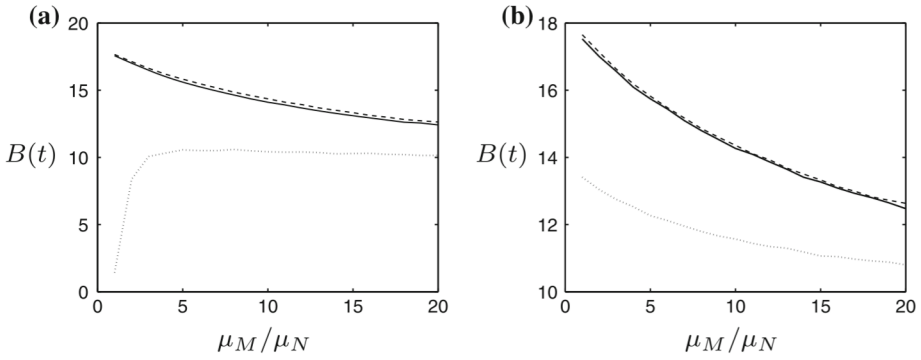

Fig. 6 Plots showing how the results of Fig. 5 depend on numerical parameters. Plots show how the vertical height of the centre of the lowest cell in the mutant patch ($B(t)$) after 20 h changes as the mutant drag ratio varies. The initial patch is a circle of radius 2, centred at $(x, y) = (W_c/2, 12)$. **a** Varying timestep (dt): *dotted*, $dt = 1/60$; *dashed*, $dt = 1/120$ (default); and *solid*, $dt = 1/240$. **b** varying steady state (T_S): *dotted*, no steady state; *dashed*, $T_S = 100$ h (default); and *solid*, $T_S = 150$ h

able to re-analyse the data in new ways allowing reuse of the stored data.

Within the framework, all simulation data is stored (rather than just the specific data needed to plot figures, or even just the figures themselves, as is usually the case); therefore, we may interrogate the simulation results to identify further points of interest. Using the existing simulation data from section "Reproducing results from Osborne et al. (2010)", we can also investigate how the number of cells in the crypt depends on the viscosity of cells for each initial mutant region height, H_M. This enables us to see where a mutation and its progeny is dominating the crypt. Figure 7 shows how the number of mutant cells in the crypt varies over time as the viscosity of mutant cells is varied.

From Fig. 7a we see that, for a mutation at the base of the crypt ($H_M = 4$), when the viscosity ratio is low ($\mu_M/\mu_N \leq 5$) the number of mutant cells increases initially, but decreases once cells are washed out of the crypt (seen in Fig. 5a, b). Additionally, as the viscosity ratio is increased, the mutant cells persist in the crypt as seen by the number of mutant cells increasing over time for $\mu_M/\mu_N \geq 10$. Moreover, once the ratio is above this value, the number of mutant cells does not increase as fast as the ratio is increased (shown by the lines becoming closer together); this is not evident from the original results published in Osborne et al. (2010). Figure 7b shows that when mutations occur towards the top of the crypt ($H_M = 12$), we see an initial increase in mutant cell number followed by a decrease (eventually to zero) for all but the largest viscosity ratios, demonstrating the wash out of mutations from the crypt. When cells are not washed out of the crypt, the number reaches a relatively steady state where the birth of cells is balanced by the removal of cells at the crypt lumen. Cells from common ancestors are dividing at similar times (as when cells divide: the two daughter cells are given a cell cycle time drawn from a normal distribution with standard deviation 1) leading to approximate waves of proliferation. In addition, due to these waves of proliferation cells are also removed approximately in groups as cells below them divide. These factors cause oscillations in mutant number to occur (see the curve for $\mu_M/\mu_N = 20$ in Fig. 7b). This suggests that the chosen cell cycle model may not be appropriate and more variation is required—something that is not evident from the original results from Osborne et al. (2010).

Running new simulations to ask new scientific questions

The final required feature of a simulation curation system is to allow new simulations to be run in a simplified manner. The study of Osborne et al. (2010) is concerned with the effect of the position of mutant cells within the crypt on whether a mutation will take over the crypt or be harmlessly removed; however, the effect of the number of mutations is not investigated. We can study this by using the model presented in Osborne et al. (2010) and varying the radius of the initial mutant population, R_M.

To do this, we consider a fixed initial height, $H_0 = 12.0$, and perform a parameter sweep over the following parameters in the model:

- $R_M \in \{0.5{:}0.5{:}3.0\}$;
- $\mu_M/\mu_N \in \{1{:}1{:}20\}$; and
- $\tau \in \{1{:}1{:}100\}$.

Again, results from 100 simulations (with varying random seed) are averaged. By using the framework proposed here, we are able to make use of the original data (which forms a subset of this investigation). Moreover, in order to perform this new experiment, we just need to change the relevant sections of the provided application. The traditional approach would require the simulation code to be modified and the results would need to be collated manually.

Figure 8 shows how the base of the mutant region varies over time as we vary the initial radius, R_M. By comparison

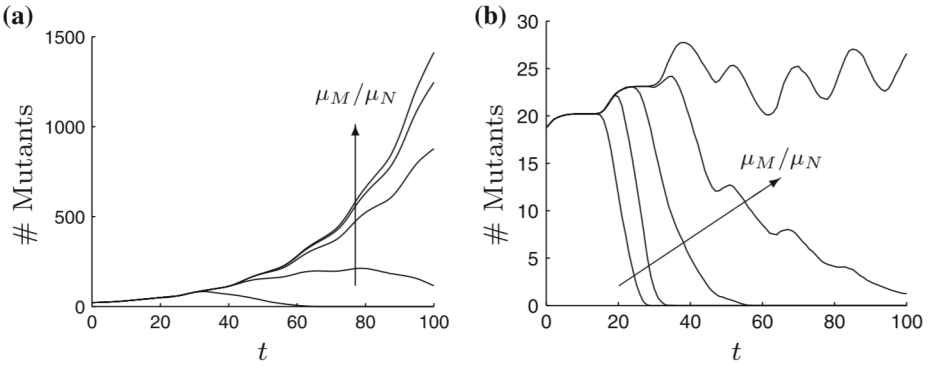

Fig. 7 Plots showing how the number of mutant cells in the crypt changes over time ($t \in [0, 100]$ h) as the initial vertical position of the mutant patch and the corresponding drag ratio vary. Results are plotted for drag ratio $\mu_M/\mu_N = 1, 5, 10, 15, 20$. The initial patch is a circle of radius 2, centred at $(x, y) = (W_c/2, H_M)$ for **a** $H_M = 4$ and **b** $H_M = 12$

of Fig. 8c with Fig. 5b, we see that increasing the radius of the initial blob has an effect similar to the mutations occurring lower in the crypt. Similarly, we also see that the behaviour for a smaller number of mutant cells (Fig. 8a) is the same as placing the mutations further towards the top of the crypt (for example, at $H_M = 16$; results not shown).

Realising the 'vision'

In section "Collaboration" we noted that there were no inherent restrictions on the sources of data that can be utilised by our approach—provided that a suitable data or file plug-in can be written to facilitate access. For example, one might envisage incorporating modelling tools such as COPASI (Hoops et al. 2006) and the Systems Biology Workbench (SBW) (Sauro et al. 2003) to enable the curation of simulations of reaction networks along with associated parameters. Here, the Systems Biology Mark Up Language (SBML) (Hucka et al. 2003) could be utilised, along with program specific details, to provide the appropriate interface. In addition, assuming the existence of appropriate interfaces, alternative multicellular modelling frameworks, such as CompuCell3D (Izaguirre et al. 2004) could be used to perform simulations.

The *data-agnosticism* of sif—by which no assumptions are made about syntax, semantics or structure of data sources—allows for a relatively straightforward means of aggregating heterogeneous data. These (and other) features of sif permitted the development of a structured data store for recording both the parameters used for a particular Chaste simulation and the results files generated by the execution of that simulation. The sif mechanism allows for these results to be generated in multiple locations by providing the necessary facilitates to achieve federation of data distributed across multiple sites.

If other groups wish to utilise sif to expose their data, it is relatively easy to do so, with the steps involved being: define a data representation for the results, store the results, provide a description of the data structure, and prescribe access via the aforementioned access control mechanism. The most important step is to define an appropriate data representation of the results—this could have enhanced utility as it also captures the information necessary to recreate the results. Once the data structure has been defined, it is a relatively simple task to produce a data plug-in for the sif middleware to allow the sharing of the data in a controlled manner—with a key feature of a data plug-in being the ability to include a data schema, which provides the name and a detailed description of the meaning of each data element.

A carefully crafted semantic definition of the data will maximise the potential for interoperability. In this way, it is possible for many groups to collaborate by generating data and informing the community at large of the semantics of that data. If additional community data sources exist, additional data plug-ins may be defined to provide access to these resources. These plug-ins might then be used in the development of more interesting queries or for comparing results of simulations against experimental data.

Going further, one might envisage communities developing their own applications that allow access to results. These application can be tailored—as the example Chaste application described in this paper was—or be more generic. For example, a more generic application is a query builder that allows a user to develop complicated distributed queries by leveraging the expressive power of SQL to achieve the desired results. The choice of style of application to develop lies with the particular group or community and the manner in which they wish to use the data.

It is recognised by several authors that there is a need to develop technique to manage the ever increasing volume of data being generated in the areas of system biology and bioinformatics. In Grossman and White (2012) the use of cloud computing—see Armbrust et al. (2010)—is considered for processing genomics, proteomic and other 'omic' data alongside the wish to combine with patient data; the authors also provide eight requirements for a 'biomedical cloud'. Work on designing cloud-based approaches for systems biology is discussed in Lansing et al. (2011); as another example, Gorton et al. (2010) considers the use of cloud for storage of data from the perspective of work flows. A discussion on the design of the design of software

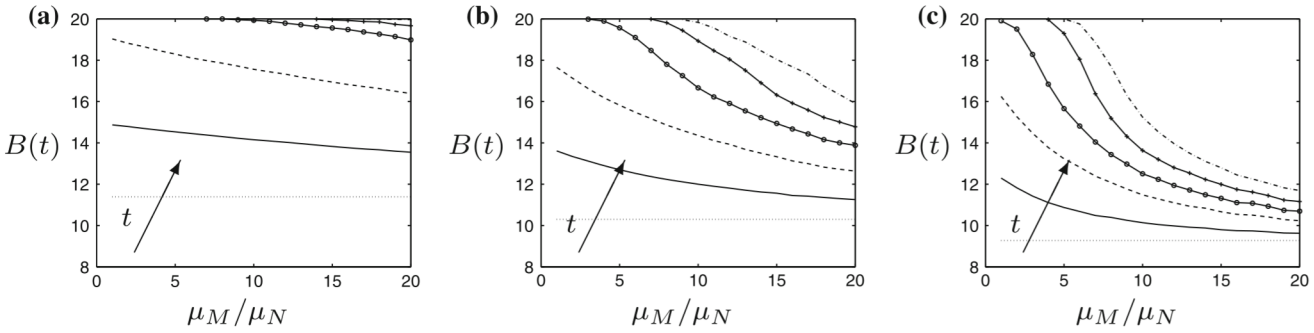

Fig. 8 Plots showing how the number of mutant cells in the crypt changes over time as the initial vertical position of the mutant patch and the corresponding drag ratio vary. Results are plotted at times $t = 0$, 10, 20, 30, 40, 50 h. The initial patch is a circle of radius R_M, centred at $(x, y) = (W_c/2, 12)$ for **a** $R_M = 1$, **b** $R_M = 2$ and $R_M = 3$

to support systems biology investigations is provided in Boyle et al. (2009)—with the key requirements postulated being the ability to develop and integrate software rapidly alongside the ability for the system to adapt rapidly to meet the next unexpected need. We plan to facilitate the integration of cloud resources, taking account of the variety of cloud concepts and APIs available, as per (Loutas et al. 2010).

Conclusions

In this paper we have described an approach, using Chaste and sif, that moves towards our aims and also addresses some of the goals expressed by Ignizio (1971, 1973) and Crowder et al. (1978). We have illustrated the power of storing simulations—both code/executables and results—in an accessible database. Such storage allows results to be validated and extended allowing the pursuit of knowledge to progress faster and enable greater scientific discoveries to be attained. Specifically, we have introduced a mechanism that records sufficient information to: permit the reproduction of results; enable the retrieval of simulation results to allow validation of claims; facilitate the extension of results by aggregating with additional results and the re-analysis of results using alternate techniques; and the running of new simulations to extend the results available for analysis. The generation of in silico experimentation results can be achieved by executing an application locally or by submitting a number of jobs via the asynchronous submission mechanism of the sif middleware. Once the experimental data has been generated, the middleware provides a mechanism allowing results from multiple data sources to be aggregated, which, in turn, can be processed to generate results and hypotheses.

The results presented in sections "Reproducing results from Osborne et al. (2010)" to "Running new simulations to ask new scientific questions" demonstrate that having access to the results allows other hypotheses to be tested, even many year later. In addition, it allows comparison of results generated by different communities—which will provide a mechanism to check the consistency of results obtained by different methods.

The storage of the parameters used for a particular experiment, along with details of the version of the code used, facilitates the process of checking that results can be reproduced—either with the original code or a more recent version. In the latter case, it can also be utilised as a method of verifying that the code behaves as expected and does not produce inconsistent or unexpected results.

The application for retrieval and processing the data could be released to the community at large to allow access to raw data and results which will facilitate independent verification of claims made in publications. Our long-term aim is that this will lead to readily available high-quality verified research data archives, which could potentially encourage the broadening of research questions being asked by allowing the composition of multiple data sources.

We have thoroughly validated the computational accuracy of results presented in Osborne et al. (2010). Moreover, we have extended the conclusions of that paper to investigate not only the position of mutations in the crypt but also the number of mutant cells, and have shown that, as well as mutant cells being washed out of or taking over the crypt, it is possible for a small number of mutant cells to persist in the crypt where proliferation is balanced by removal. An interesting avenue for further study is the stability of this state: what would be required to move from this steady state to cells taking over the crypt? By undertaking new simulations using the same framework as Osborne et al. (2010), we have also shown that the number of mutations is as important as position of mutations in determining whether a mutation takes over the crypt.

In this particular example, the current job submission and query application is tightly coupled with the example Chaste application. While this is reasonable for a proof of principle, it would be more useful for there to be a more generic application. With this in mind, a longer term aim is to develop an application framework that allows for the rapid

customisation of the inputs necessary for a given algorithm plug-in. This can be achieved by the use of a parameter file which describes the types of input necessary. The parameter file could also indicate the ordering of the different inputs and the number of panes to distribute them over. A suitable initial candidate for the parameter file is the plug-in definition file, which provides all the necessary information, with standardised methods for rendering the data input screen and generating the request from the inputs. A possible approach to provide a framework to cope with the separate concerns in the prototype application is to factor each concern out into separate applications; a query application could then use the data definitions of each of the data plug-ins it wished to use to provide a list of fields and potential joins. This approach has already been tested in a portal application, which presents the users with a simplified query-building mechanism which involves selecting which tables and fields to use and defining restrictions on field values if needed.

Whilst it is important to adopt standards for data and exchange formats when considering things like pathway data—see Cary et al. (2005)—we consider flexibility in data definition to be essential when capturing experimental data. This would be difficult to achieve if a rigid data structure is prescribed. Although there may be a core data set that is common to a particular class of experiments, there are likely to be differences introduced by the particular focus of each individual experiment. The use of a 'bottom-up' approach—as considered in Tromans et al. (2008)—allows for each experimental group to define their own data structure for capturing their generated data. If this data is to be useful to others, then the group generating the data needs to provide a semantic meaning to each data element. In this way, data from complementary experiments generated by different groups could be federated in a meaningful way. In Jenkinson et al. (2008), a distributed data annotation system is considered; the approach uses an XML data representation and a limited number of commands help to facilitate user-driven data integration via graphical interfaces. While the authors note that approaches based on more complex middleware—such described in this contribution—offer benefits such as more complex query capabilities, we acknowledge that a 'middle path' will be necessary if our longer term aims are to be met.

Acknowledgments The authors acknowledge financial support from the BBSRC, through the OCISB project (BB/D020190/1). JMO is supported by Microsoft Research Cambridge and the EPSRC funded University of Oxford Doctoral Training Centres.

References

Alberts B, Johnson A, Lewis J, Raff M, Roberts K, Walter P (2002) Molecular biology of the cell, 4th edn. Garland Science, New York

Anderson AE, Ellis BJ, Weiss JA (2007) Verification, validation and sensitivity studies in computational biomechanics. Comput Methods Biomech Biomed Eng 10(3):171–184

Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, Zaharia M (2010) A view of cloud computing. Communications of the ACM 53:50–58

Boyle J, Rovira H, Cavnor C, Burdick D, Killcoyne S, Shmulevich I (2009) Adaptable data management for systems biology investigations. BMC Bioinf 10(1):79

Brady JM, Gavaghan DJ, Simpson AC, Parada MM, Highnam RP (2003) e-DiaMoND: a grid-enabled federated database of annotated mammograms. In: Berman F, Fox GC, Hey AJG (eds) Grid computing: making the global infrastructure a reality. Wiley, Chichester, UK, pp 923–943

Brambilla M, Ceri S, Passamani M, Riccio A (2004) Managing asynchronous web services interactions. In: Proceedings of the IEEE international conference on web services (ICWS'04), pp 80–87

Buske P, Galle J, Barker N, Aust G, Clevers H, Loeffler M (2011) A comprehensive model of the spatio-temporal stem cell and tissue organisation in the intestinal crypt. PLoS Comput Biol 7(1):e1001045

Cary MP, Bader GD, Sander C (2005) Pathway information for systems biology. FEBS Lett 579(8):1815–1820

Crowder HP, Dembo RS, Mulvey JM (1978) Reporting computational experiments in mathematical programming. Math Program 15:316–329. doi:10.1007/BF01609036

Di Ventura B, Lemerle C, Michalodimitrakis K, Serrano L (2006) From in vivo to in silico biology and back. Nature 443(7111):527–533

Gorton I, Liu Y, Yin J (2010) Exploring architecture options for a federated, cloud-based system biology knowledgebase. In: 2010 IEEE second international conference on cloud computing technology and science (CloudCom), pp 218–225

Greenwood M, Goble C, Stevens R, Zhao J, Addis M, Marvin D, Moreau L, Oinn T (2003) Provenance of e-Science experiments—experience from bioinformatics. In: The UK OST e-Science second all hands meeting 2003 (AHM'03), pp 223–226

Grossman RL, White KP (2012) A vision for a biomedical cloud. J Intern Med 271(2):122–130

Hoops S, Sahle S, Gauges R, Lee C, Pahle J, Simus N, Singhal M, Xu L, Mendes P, Kummer U (2006) Copasi—a complex pathway simulator. Bioinformatics 22(24):3067–3074

Hucka M, Finney A, Sauro H, Bolouri H, Doyle J, Kitano H, Arkin A, Bornstein B, Bray D, Cornish-Bowden A et al (2003) The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models. Bioinformatics 19(4):524–531

Ignizio JP (1971) On the establishment of standards for comparing algorithm performance. Interfaces 2(1):8–11

Ignizio JP (1973) Validating claims for algorithms proposed for publication. Oper Res 21(3):852–854

Izaguirre J, Chaturvedi R, Huang C, Cickovski T, Coffland J, Thomas G, Forgacs G, Alber M, Hentschel G, Newman S et al (2004) Compucell, a multi-model framework for simulation of morphogenesis. Bioinformatics 20(7):1129–1137

Jenkinson A, Albrecht M, Birney E, Blankenburg H, Down T, Finn R, Hermjakob H, Hubbard T, Jimenez R, Jones P, Kahari A, Kulesha E, Macias J, Reeves G, Prlic A (2008) Integrating biological data the distributed annotation system. BMC Bioinf 9(Suppl 8):S3

Kitano H (2002) Computational systems biology. Nature 420(6912):206–210

Lansing C, Liu Y, Yin J, Corrigan A, Guillen Z, van Dam K, Gorton I (2011) Designing the cloud-based DOE systems biology knowledgebase. In: 2011 IEEE international symposium on parallel and distributed processing workshops and Phd forum (IPDPSW), pp 1062–1071

Loeffler M, Stein R, Wichmann H, Potten C, Kaur P, Chwalinski S (1986) Intestinal cell proliferation. i. a comprehensive model of steady-state proliferation in the crypt. Cell Prolif 19(6):627–645

Loutas N, Peristeras V, Bouras T, Kamateri E, Zeginis D, Tarabanis K (2010) Towards a reference architecture for semantically interoperable clouds. In: 2010 IEEE second international conference on cloud computing technology and science (CloudCom), pp 143–150

Meineke FA, Potten CS, Loeffler M (2001) Cell migration and organisation in the intestinal crypt using a lattice-free model. Cell Prolif 34:253–266

Osborne J, Walter A, Kershaw S, Mirams G, Fletcher A, Pathmanathan P, Gavaghan D, Jensen O, Maini P, Byrne H (2010) A hybrid approach to multi-scale modelling of cancer. Phil Trans R Soc A 368:5013–5028

Pathmanathan P, Bernabeu M, Bordas R, Cooper J, Garny A, Pitt-Francis J, Whiteley J, Gavaghan D (2010) A numerical guide to the solution of the bidomain equations of cardiac electrophysiology. Prog Biophys Mol Biol 102(2–3):136–155

Pitt-Francis J, Bernabeu M, Cooper J, Garny A, Momtahan L, Osborne J, Pathmanathan P, Rodriguez B, Whiteley J, Gavaghan D (2008) Chaste: using agile programming techniques to develop computational biology software. Phil Trans R Soc A 366:3111–3136

Pitt-Francis J, Pathmanathan P, Bernabeu M, Bordas R, Cooper J, Fletcher A, Mirams G, Murray P, Osborne J, Walter A, Chapman S, Garny A, van Leeuwen I, Maini P, Rodriguez B, Waters S, Whiteley J, Byrne H, Gavaghan D (2009) Chaste: a test-driven approach to software development for biological modelling. Comp Phys Comm 180(12):2452–2471

Radetzki U, Cremers A (2004) IRIS: a framework for mediator-based composition of service-oriented software. In: Proceedings of the IEEE international conference on web services, 2004 (ICWS'04), pp 52–756

Sauro H, Hucka M, Finney A, Wellock C, Bolouri H, Doyle J, Kitano H (2003) Next generation simulation tools: the systems biology workbench and biospice integration. Omics 7(4):355–372

Simpson AC, Power DJ, Russell D, Slaymaker MA, Bailey V, Tromans CE, Brady JM, Tarassenko L (2010) GIMI: the past, the present, the future. Phil Trans R Soc A Math Phys Eng Sci 368:3891–3905

Simpson AC, Slaymaker MA, Gavaghan DJ (2010b) On the secure sharing and aggregation of data to support systems biology research. In: Proceedings of the 7th international conference on data integration in the life sciences (DILS 2010). Springer Lecture Notes in Computer Science, vol 6254, pp 58–73

Slaymaker MA, Power DJ, Russell D, Simpson AC (2008a) On the facilitation of fine-grained access to distributed healthcare data. In: Jonker W, Petkovic M (eds) Proceedings of secure data management (SDM) 2008. Springer Lecture Notes in Computer Science, vol 5159, pp 169–184

Slaymaker MA, Power DJ, Russell D, Wilson G, Simpson AC (2008b) Accessing and aggregating legacy data sources for healthcare research, delivery and training. In: Wainwright RL, Haddad H (eds) Proceedings of the 2008 ACM symposium on applied computing (SAC), pp 1317–1324

Tromans CE, Brady JM, Power DJ, Slaymaker MA, Russell D, Simpson AC (2008) The application of a service-oriented infrastructure to support medical research in mammography. In: Proceedings of MICCAI-Grid 2008

Van Leeuwen I, Mirams G, Walter A, Fletcher A, Murray P, Osborne J, Varma S, Young S, Cooper J, Doyle B, Pitt-Francis J, Momtahan L, Pathmanathan P, Whiteley J, Chapman S, Gavaghan D, Jensen O, King J, Maini P, Waters S, Byrne H (2009) An integrative computational model for intestinal tissue renewal. Cell Prolif 42:617–636

Zdun U, Voelter M, Kircher M (2004) Pattern-based design of an asynchronous invocation framework for web services. Int J Web Serv Res 1