Nat Comput (2015) 14:63–82
DOI 10.1007/s11047-014-9468-z

Principled simulation of cell proliferation dynamics using the CoSMoS approach

Fiona Polack · Alastair Droop

Published online: 17 October 2014

Abstract A collaboration between cancer biologists and academic software engineers has been exploring the development of an agent-based simulator to inform and support work on the dynamics of cell proliferation in the study of prostate disorders. The research has influenced and been informed by the CoSMoS project. This paper presents the simulation project (which is not yet complete). We reflect on the reality of following CoSMoS principles; we describe the domain exploration and show how software modelling approaches (here, Petri nets, state diagrams) can be used to express both biological and software models. We explore fitness for purpose and consider ways to present a fitness argument. We consider issues in choosing simulation media and mapping from domain models through to code. The implementation emphasis is on traceability to support reuse and extension of the simulator, as well as demonstrable fitness for purpose. Initial work on calibration is presented. We discuss the calibration results, that both support and challenge the design and assumptions captured in the domain modelling and development activities.

Keywords Cell proliferation · Agent-based simulation · CoSMoS · Modelling and simulation

F. Polack (✉)
YCCSA/Department of Computer Science, University of York, York YO10 5DD, UK
e-mail: fiona.polack@york.ac.uk

A. Droop
Cancer Research UK Leeds Centre, University of Leeds, Leeds, UK

1 Introduction

The CoSMoS principled approach to complex systems modelling and simulation (Andrews et al. 2010; 2011) has been used in contexts as diverse as immune infection and response, (Albergante et al. 2013; Alden 2012; Alden et al. 2012; Greaves et al. 2013; Moore et al. 2013; Read et al. 2009a, b, 2012; Read 2011; Williams et al. 2013), plant auxin transport (Garnett et al. 2008, 2010a, b), and social-ecological modelling (Forrester et al. 2012). However, CoSMoS is not complete and immutable. Each project uses CoSMoS guidelines in different ways and to different effect. Each project throws up new challenges, but also new ways to manage and benefit from collaborative simulation-based research.

This paper focuses on CoSMoS use in developing a cell proliferation simulator. The project has been ongoing for four years, and provides new insights into the CoSMoS approach. The project seeks to model and simulate prostate epithelial cell proliferation dynamics, supporting exploration and generation of research hypotheses for both normal and anomalous prostate characteristics.

Cell proliferation requires a cell to divide, with a complex duplication process that may introduce faults into one or both resulting cells. Such anomalies in cell proliferation are implicated in the origins of many diseases and disorders. Our project focuses on two such prostate conditions: prostate cancer (PC) and benign prostatic hyperplasia (BPH) (Droop et al. 2011; Polack et al. 2011). Biological understanding of the dynamic origins of these conditions is largely based on observation of tissue samples from biopsies stained so that cells of different types can be counted. Data is limited to what can be counted, and interpretation relies on hypotheses of potential cell dynamics.

Whereas many of the papers cited above focus on biological modelling and outcomes of CoSMoS simulation projects (capturing the domain and expected behaviours; identifying and exploring biological hypotheses; arguing the fitness for purpose of biological and results models), this paper is focused mostly on the proliferation study as a vehicle for exploring the software engineering aspects of CoSMoS.

Section 2 briefly introduces the CoSMoS approach, and discusses how the CoSMoS concepts relate to the prostate cell proliferation simulation study. Section 3 introduces the software engineering concept of roles, describing the responsibilities of the roles and formation of the collaboration at the heart of the project. After this, the sections follow the sequence of CoSMoS products—noting that the results model is out of scope of this paper, and the research context is discussed in the context of the other products. Section 4 introduces the cell proliferation domain, with examples of typical domain information and data. Domain assumptions and their recording is explored. The domain exploration is summarised in a stylised hypothesis diagram. Section 5 focuses on the development of a domain model that captures the relevant concepts for a simulator and establishes the scope and *purpose* of the simulation exercise. The domain model must faithfully capture the domain biology, and is labelled using biological terminology; it should not express computational concepts (computational data types and structures, computer functions, etc.). However, the domain model is also the most abstract of the engineering models that are typically used in software development, and uses a well-founded software engineering notation: the software development will proceed by translating the biological terminology into computing terminology and successively elaborating the modelled software components until code can be written. This section (Sect. 5) introduces this aspect of software engineering development and issues raised concerning seamlessness and traceability. Domain model assumptions are considered. The section concludes with a discussion of fitness for purpose, its importance and how it can be established and documented. Section 6 discusses the CoSMoS platform model, which represents the computational design for the simulator. The cell proliferation platform design has been the focus of a range of software engineering research activities, which are summarised here. Section 7 outlines prototype implementations created for the simulator, summarising aspects such as language suitability and software engineering support which are not normally explored in simulation papers. The section exemplifies the value of working within a collaborative framework such as CoSMoS for a project that does not have continuity of personnel. Section 7 also considers calibration of the simulator, showing how calibration can lead to revisiting of assumptions and design decisions—and, here, to the next iteration of the cell proliferation project. Section 8 summarises the key domain and software engineering outcomes of the project to date.

2 Using the CoSMoS approach for cell proliferation

CoSMoS is a principled approach to complex systems modelling and simulation, which includes general advice on project structure and lifecycle, domain exploration, collaboration development, software engineering, and results analysis. This section briefly introduces the CoSMoS approach, and reflects on the reality of the phases outlined by CoSMoS (Fig. 1). More detail appears in other papers in this Special Issue (e.g. Polack 2014). Background and rationale for CoSMoS has been widely reported (Andrews et al. 2011; Polack 2010, 2012; Polack et al. 2008, 2009, 2010), building on the first draft process, (Andrews et al. 2010).

A natural side-effect of the first iteration of the *discovery* phase (Fig. 1) is to establish a collaboration: the project participants meet and develop working relationships. In the prostate cell proliferation study, as in many applications of CoSMoS, a kick-off meeting brought together members of the laboratory working on prostate disorders and computer scientists who were working on the software engineering aspects of complex-systems simulation. The meeting explored how a computer simulation might be used to address hypotheses concerning the effect of cell proliferation dynamics that could not be completely addressed by laboratory research. From this meeting, a core collaboration team was established, comprising three academic software engineers and three prostate cancer biologists, one of whom had some software engineering experience. Subsequent meetings were effectively tutorials in prostate cell biology, and helped to develop mutual understanding and trust.

As in other projects, the discovery phase was revisited many times. As a project progresses, the need for additional biological background arises often, with new or more-focused questions being posed. The initial establishment of a trusting relationship is critical to successful collaboration. Here, it has been particularly beneficial, as the work has no long-term funding or dedicated research staff to maintain day-to-day working relationships. Each new researcher has had to be taken on a voyage of discovery, and each new briefing improves collective understanding of the domain.

The objective of the *development* phase (Fig. 1) is to create a fit-for-purpose simulator, using a systematic, traceable software engineering approach. CoSMoS specifically advocates traceable software engineering to promote

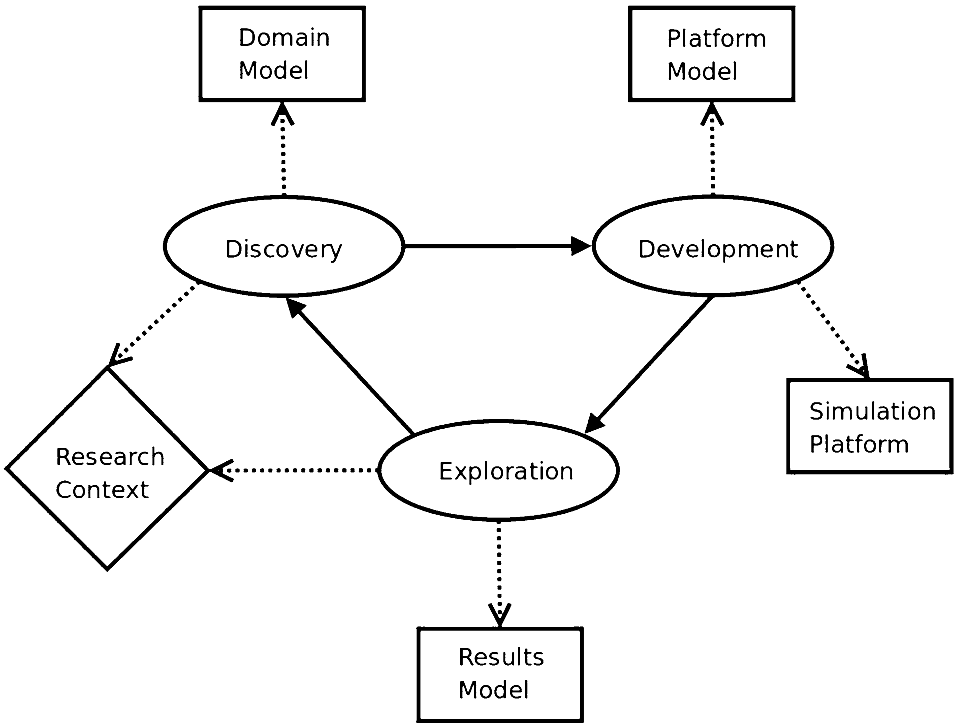

Fig. 1 Phases of the CoSMoS process: phases are shown as *ellipses* and related products as *boxes* (Andrews et al. 2010; Stepney 2012). In this case study, the CoSMoS process started with the Discovery phase. The phases are iterated many times in the course of a simulation project

code maintainability (including extension and reuse) and to facilitate demonstration of fitness for purpose (Polack et al. 2009). Software engineering typically aims to create an application to meet a set of objective requirements, by developing an abstract model of a system that meets the requirements. Development systematically maps abstract concepts to design concepts and ultimately to code. Traceability means that concepts at any level can be mapped to and from concepts at adjacent levels—so, the concept of a prostate epithelial cell in the abstract (in software engineering terms) domain model maps through successive design steps to a specific part of the code that implements the simulator.

In the *exploration* phase (Fig. 1), the focus is on the use of the computer simulation—i.e., exploring the dynamics of cell proliferation. The first part of exploration is to calibrate the simulator against biological examples. Once the collaborators are confident of the simulator performance, we will then explore hypotheses of the neogenesis, growth and treatment of the targeted prostate conditions, using the simulation results to inform and target the laboratory research. To date, we have completed initial calibration runs and are following up on questions raised by calibration (see Sect. 7). The prompting of further iterations of the discovery phase is common, and the nature of a CoSMoS-style collaboration facilitates this. Here, the ongoing revisiting of the discovery phase is itself driving laboratory research on cell staining and counting by the domain experts. Some of the key questions that have led to new biological research are:

– how to stain different sample tissues in order to efficiently count cells of the various types; what cell types can and cannot be counted, and at what level of accuracy;
– how to distinguish normal prostate data, when most of our biopsy samples come from patients in the process of being diagnosed with BPH;
– how to count dead cells when only the process of dying is detectable.

Figure 1 shows, in addition to CoSMoS phases, the series of *products* that CoSMoS proposes as a development lifecycle. The presentation of the cell proliferation case study that follows is structured according to these products. First, we introduce the participants and their roles in the collaboration.

3 The research team and roles

CoSMoS advocates the use of roles, reflecting the key stakeholders in the project (Polack et al. 2010). Roles help to clearly define project responsibilities (Andrews et al. 2010). In software engineering, a role may be taken by many people, and a person may take many roles. Two principal roles are required:

– the role of *domain expert*: here, taken by the staff of the Maitland Lab at the University of York, led by Prof. Norman Maitland. The project originated in an initiative by Dr Alastair Droop, a researcher in the team who had connections with the CoSMoS researchers in the York Centre for Complex Systems Analysis (YCCSA) through previous project work;
– the role of *developer*: here, taken by software engineers based in YCCSA, whose interest is in supporting and developing a principled approach to the simulation of complex systems, notably the use of agent-based models for simulation. The ongoing development is led by Dr Fiona Polack.

These roles encompass the colleagues, students and interns working in the host research groups. Roles relate to the phases and products of the project. The domain expert role is responsible for the cell proliferation and prostate information, and defines the interpretation of cell biology that is relevant to the project—ensuring that those playing a developer role can focus on software engineering, rather than biological debate. However, the domain expert must also have oversight of the work of the developers, in order to validate the model of the domain and its preparation for use in software engineering.

The developer role is responsible for the quality of the software, and for demonstrating that the software effectively implements the model of the biological domain. The first developer activity is to create abstract software engineering models of the relevant aspects of the domain in such a way that the domain expert can understand how the domain concepts are being captured: it is not the biologists' role to develop software, but it *is* the domain expert's role to confirm that the software development started from the right place. In exploration, the domain expert leads in interpretation and hypothesis generation, but the developer role is responsible for maintaining and adapting software, in calibration and subsequent experimentation activities.

Managing the diverse and continually changing project team has been challenging, but the principled CoSMoS approach and emphasis on recording justification and rationale has helped to achieve continuity. This paper includes reflection on some of the lessons learnt about the reality of seamless development from a real domain to computer code.

4 The prostate domain

This section introduces the domain, the prostate cell populations and their context. The information presented here is chosen to be illustrative of the sort of domain material that is typically accumulated by a collaborative team, rather than to give a comprehensive introduction to prostate cell proliferation.

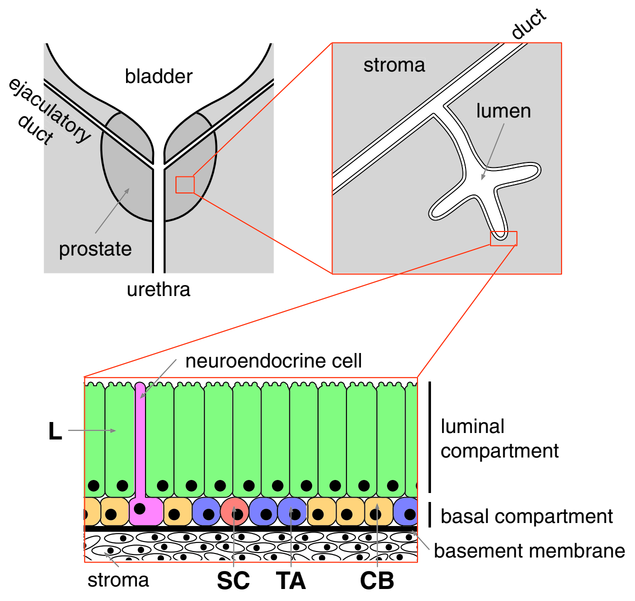

Fig. 2 Cartoon showing the context for the prostate epithelial cells of interest: luminal cells (L) in the luminal compartment; stem cells (SC), transit-amplifying cells (TA) and committed basal cells (CB) in the basal compartment. See text for relevant details

The context for the simulation project is epithelial cell proliferation and disorders characterised by aberrant cell proliferation. Prostate disorders such as PC and BPH affect most men in old age (Maitland and Collins 2008). PC can be fatal. The clinical symptoms of BPH are debilitating, characterised by an enlarged prostate that constricts urine flow. Surgery (where possible) or drug treatment typically relieve the symptoms for a few years, but do not "cure" the conditions. Typical drug treatment for PC involves androgen modulation therapy, often resulting in transient tumour bulk decline; but androgen-insensitive disease eventually develops, which is usually untreatable and fatal (Maitland 2013).

The normal prostate is a walnut-sized organ, though it can expand significantly through conditions such as BPH and PC. The organ is responsible for producing the secretions that convey semen. The organ is relatively self-contained (in biological terms) and forms an unusually precise subject for modelling. The prostate surrounds the urinary tract, and may constrict flow of urine (Fig. 2, upper left). The ducts of the prostate form clusters of voids, or lumen (Fig. 2, upper right), and the cells of interest are the epithelial cells that surround the lumen, specifically basal and luminal cells (Fig. 2, lower). The simulation project is not concerned with the stromal cells that form the bulk of the prostate, nor with neuroendocrine cells.

Basal and luminal cells (Fig. 2, lower) can be separately identified in cell culture and by staining of biopsy slices. It is biologically meaningful to identify three phases for basal cells: *stem cells*, which may proliferate (by differentiation with or without further division) to *transit amplifying cells*, which in turn give rise to *committed basal cells*. The committed basal cells may then differentiate to luminal cells.

Figure 2 is typical of the information presented as part of a biological domain. The sketches are backed up by extensive evidence from the domain experts' research, knowledge and interpretation of the domain literature. Whilst much of this information becomes available through the initial collaboration meetings, there is a continual interaction, in which developers raise new questions and domain experts provide new answers or interpretation to guide the development of a fit-for-purpose simulation.

Cell proliferation can be studied at various levels of structural and biochemical detail, but, for this project, the focus is on the dynamics of proliferation, as expressed through epithelial cell division and differentiation processes. There is biological debate over exactly which hypothesised division and differentiation pathways exist and the rates of proliferation (Droop et al. 2011). Prostate cell dynamics are hard to study. Domain experts have assembled evidence from literature, from which normal and cancerous prostate epithelial cell ratios can be estimated – but the published information identified to date is from very small cell counts. To improve on the published data, the Maitland Lab is collecting cell counts from biopsy samples from patients. The data limitations are quite challenging. Currently, 35 tissue slides have been analysed, from 10 patients, totalling around 3000 prostate epithelial cells (the domain expert believes this is a significantly larger sample than for any currently published studies). The biopsy samples are from patients diagnosed with BPH, but not all slides show characteristics of BPH: some samples show either normal or PC characteristics. The sample data are from single biopsy samples per patient, and do not even give a representative picture of the state of one whole prostate at one point in time. There is no time-series data, as it is very rare for a patient to have more than one prostate biopsy, and even rarer to have permission for their scientific use. We do not have other clinical information, so we cannot order the separate patient samples into any sort of disease-related time ordering. In our samples, each slide has between 24 and 505 distinguishable epithelial cell nuclei. Stem cells form at most 1 % of the epithelial cells. The ratio of basal cells (Fig. 2: SC, TA and CB combined) to luminal cells ranges from 57:8 (7.125) to 7:12 (0.583).

Using the biopsy sample data plus review of published prostate cell studies, the domain experts suggest that a representative simulation would have similar numbers of basal and luminal cells, and that the number of stem cells would be at most about 1 % of the total number of cells. The domain experts would expect a prostate showing the characteristics of BPH to have similar cell ratios, but a significant increase in the overall number of cells. Similarly, a prostate showing the characteristics of PC should have significantly more (in extreme cases, 90 % of all cells) luminal cells.

The cell proliferation simulation is motivated by the need to explore cell dynamics of a normal prostate and the variations from normal dynamics that give rise to the cell ratios characteristic of PC or BPH. The long-term goal is to be able to use the simulator to show how these prostate conditions might originate, and to understand why and how the conditions might re-emerge some time after surgery or existing drug treatments by mimicking the changes to cell dynamics that the current treatments are believed to cause (e.g. surgery removes predominantly luminal cells and reduces the total number of cells). The overarching hypothesis is that, by identifying aberrant proliferation dynamics, it will be possible to target treatments on eradication or prevention rather than on the temporary control of the symptoms.

A principle of the CoSMoS approach is that the designated domain expert provides and interprets information and data about the domain. This point is important: there has traditionally been scepticism over the scientific use of simulators where developers have not worked closely with domain experts, or have not documented the domain evidence used to inform the development. In a collaborative CoSMoS project, developers are not responsible for discovering or interpreting biological information, but they do need some domain understanding to facilitate intelligent discussion with the domain experts and to enable selection and abstract modelling of relevant domain concepts.

4.1 Domain assumptions

Domain exploration includes consideration of the feasibility, scope and scale of simulation. Assumptions and justifications are recorded as part of the *research context* (Fig. 1), which is used to contextualise simulation results. Domain assumptions may be identified at any point in a simulation development. It is not always obvious when an assumption is being made, so assumptions are never complete, and what is recorded may appear somewhat arbitrary. The following commentary considers some of the domain assumptions that were identified in the exploration phases of the cell proliferation project.

1. We can treat epithelial cells as falling into one of four types: stem cells can differentiate into transit amplifying cells, which can differentiate into committed

basal cells, and finally to luminal cells. Stem cells, committed basal cells and transit amplifying cells are forms of basal cell, with stem cells being the source of regeneration to replace dead cells.

– Reason: The domain expert identified stages from the continuum from a stem-cell phenotype to a luminal phenotype that it would be useful to simulate. Initially, it was agreed to use four cell types with distinct expression of surface markers. Subsequently, problems with staining and counting led the domain expert to propose refocusing on stem, other basal, and luminal cells.
– Justification: The simulator needs to represent concepts that are traceable to biological data. The chosen abstract cell types represent cells that can be biologically distinguished due to their biochemical behaviour, to provide appropriate data on cell counts or ratios for input to the simulator and for results comparison.

2. The model of cell proliferation can focus on the epithelial cells. The domain expert notes that, whilst blood vessel formation (via the stroma) and hormonal interactions (via neuroendocrine cells) are known to be important in conditions such as PC, these can be ignored.

   – Reason: The simulator has to simplify and abstract some domain detail so that it is computationally tractable and simple enough to interpret against the biological research. Although stromal and neuroendocrine cells are involved, the domain experts do not have data on these cells.
   – Consequence: The simulation will not be appropriate for study of any prostate conditions that are heavily influenced by blood supply, or any condition in which stromal or neuroendocrine cells are implicated.

3. Spatial aspects such as cell positions and cell contact can be ignored.

   – Reason: Spatial aspects such as crowding and proximity are implicated in, for instance, cell death and cell-cell signalling. However, the spatial aspect was not well understood, and its omission simplified the model and implementation. We return to spatial assumptions in Sect. 7.
   – Consequence: We need to design an alternative to crowding as a control mechanism – for instance through feedback between cell death and division events, and limits on cell numbers. We need to ensure that simulation of the effect of environmental influences includes those that, in reality, have an effect via spatial aspects.

4. The cell division process is expressed as simple cell duplication, followed by options for differentiation.

   – Justification: In reality, a cell usually first replicates its genome, then divides to accommodate each new genome. However, for the purpose and level of abstraction used in the simulator, there is no functionality associated with the process of division. We are only interested in the abstraction to probabilities of division and differentiation, and in the dynamic effect of mutating probabilities.
   – Consequences: Representing division as a single process means that we cannot add genome-level influences directly. Cell division in the simulator needs to represent an appropriate abstraction of heritable genomic information, and needs to represent the possibility of faulty genome copying (mutation).

5. It is assumed that we can ignore the genes that control cell behaviour in response to extra-cellular environmental influences.

   – Reason: Intra-cellular mechanisms are better studied in the laboratory, or at a lower level of abstraction than we consider here.
   – Justification: The full set of environmental influences and the consequent intra-cellular behaviours are not known, and are not of relevance here, as we are only interested in environmental influences that cause acceleration or deceleration of cell division and differentiation.
   – Consequences: The simulator cannot be used to explore detail of different environmental influences.

4.2 Collating the domain discoveries

In the CoSMoS approach, the accumulating information about the domain needs to be assimilated and co-ordinated so that the purpose and scope of simulation can be agreed among the collaborators. After experimenting with various informal models, CoSMoS researchers arrived at a domain *hypothesis model* (or, expected behaviours model). The hypothesis model summarises what is observable in the domain, the hypotheses that the domain experts would like to explore, and the real concepts that are (probably) involved in behaviours that need to be simulated in order to explore the hypotheses.

A hypothesis model is most meaningful within the context of its collaboration, and may take many hours to create, typically sketched on a large whiteboard. The example shown in Fig. 3 was created retrospectively for the prostate cell proliferation domain, and has had to be compressed to fit the page format. Normally, these

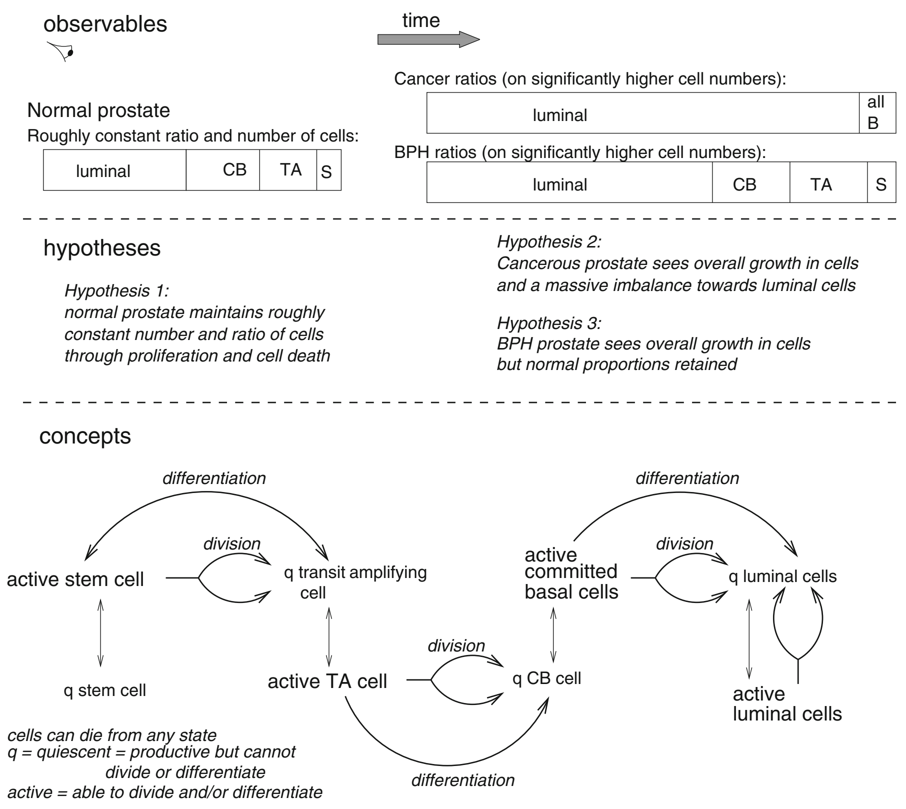
observables
time
Normal prostate
Roughly constant ratio and number of cells:
luminal
CB
TA
S
Cancer ratios (on significantly higher cell numbers):
luminal
all B
BPH ratios (on significantly higher cell numbers):
luminal
CB
TA
S
hypotheses
Hypothesis 1: normal prostate maintains roughly constant number and ratio of cells through proliferation and cell death
Hypothesis 2: Cancerous prostate sees overall growth in cells and a massive imbalance towards luminal cells
Hypothesis 3: BPH prostate sees overall growth in cells but normal proportions retained
concepts
differentiation
division
active stem cell
q transit amplifying cell
q stem cell
active TA cell
division
q CB cell
differentiation
active committed basal cells
differentiation
division
q luminal cells
active luminal cells
cells can die from any state
q = quiescent = productive but cannot divide or differentiate
active = able to divide and/or differentiate

Fig. 3 Simplified hypothesis diagram created during cell proliferation domain exploration, showing observable phenomena (*top*), hypotheses (*middle*) and concepts (*bottom*) of relevance to cell proliferation dynamics. The original diagram is a white-board sketch that cannot be easily included here

diagrams are much less stylised than the reproduction here implies.

The top section of the hypothesis diagram captures observable phenomena from reality. In Fig. 3, the information on observables in the top panel of the diagram summarises characteristic cell proportions observed in the prostate. The time arrow shows that, in the domain of interest, the observable phenomena evolve over time. On the left of Fig. 3, a divided rectangle shows approximate ratios of epithelial cell types in the normal prostate. On the top-right of Fig. 3, two divided rectangles characterise the observable epithelial cell proportions in PC (upper) and BPH (lower). The divided rectangles are a visual expression of cell ratios; here, they are not intended to express the spatial aspects (shown in biological cartoons such as Fig. 2), because of the agreed domain assumption that spatial aspects can be ignored (Sect. 4.1).

The middle panel of the hypothesis diagram states the hypotheses that the domain experts wish to explore. In Fig. 3, separate hypotheses are identified for normal, PC and BPH cell dynamics.

The bottom panel of the hypothesis diagram captures biological concepts that are to be the focus of the simulation project. The diagram does not replace the biological information recorded in the research context—for instance, in Fig. 3, the diagram does not give details of cell ratios or numbers.

The hypothesis diagram for the cell proliferation domain captures two levels of behaviour. The higher-level behaviour (curved horizontal arrows) represents cells of a particular type differentiating or dividing. At the lower level, individual cells can change state from active to quiescent (straight vertical arrows). Here, *quiescent* applies to any cell in a state which precludes division or differentiation. Most cells spend as much as 80 % of the time in a quiescent state, whilst luminal cells are almost always quiescent, only dividing in rare pathological situations. Both levels of behaviour in Fig. 3 are simplifications, but domain experts believe that the abstraction is sufficient to be able to explore patterns of cell dynamics.

From the initial domain exploration, and before the domain model is produced, the CoSMoS approach strongly encourages agreement on a *purpose* for the simulator. The purpose helps to focus activity and to determine the appropriate scope and level for the simulator, as well as to guide the design of simulator outputs. Here, the agreed purpose for the simulator (Polack et al. 2011) is to test division and differentiation rates in order to establish candidate dynamics for development and treatments of prostate disorders, summarised as follows.

1. To develop a model of cell differentiation and division, based on prostate cell populations from laboratory research. The purpose of the first phase is to establish

the parametrisation and credibility of the simulator by replicating observed cell population dynamics, represented as changing proportions of cells in a “normal” prostate.

2. Building on the simulation of a “normal” prostate, to develop simulations that capture the effect on cell proliferation of known environmental variation and mutation. The purpose of the model is to explore the emergence of cell proportions indicative of specific prostate conditions (PC, BPH).
3. Using these models of normal and abnormal prostate cell dynamics, to develop simulation experiments that can be used to test biological hypotheses for the development and control of the specific prostate conditions.

5 Domain model development

Once exploration of the domain has established a purpose for simulation, developers can start the development. The domain model forms a bridge between the domain scientist’s view of the domain and the software engineering development process, and is part of the CoSMoS development phase (Fig. 1). This section describes and discusses the way in which domain modelling moves from a collection of biological information to a domain model that forms the starting point for a traceable, seamless development. As well as considering domain modelling and domain modelling assumptions, we consider fitness for purpose, showing how an argument can be created to express the confidence of the collaborating team in this abstract design.

The biological domain illustrations such as Fig. 2 are informal: the symbols in the picture, or cartoon, are not intended to convey a formal or consistent meaning. Similarly, a hypothesis diagram such as Fig. 3 simply provides a visual summary. In order to support systematic software engineering development of a simulator that is demonstrably fit for purpose, the relevant domain information needs to be expressed in abstract models with well-defined notations, and then successively refined to code. Each concept should be clearly traceable through the series of models. If the domain model changes, it should be straightforward to trace through to the parts of the simulator code that need to change. This is referred to as *seamless* development or *traceability*.

Domain modelling is led by software engineers from the developer team, whereas domain experts lead the domain discovery phase. The domain experts need to manage the information about the domain, using their expertise to determine what is relevant, and interpreting as needed for the software engineers who lack the deep contextual knowledge of the scientists. The software engineers raise questions in order to test and confirm their understanding of the domain experts’ perspective on the domain. To achieve demonstrable fitness for purpose, thought needs to be given to traceability between the biological domain and the software engineering domain model: this step is always a point of weakness in software engineering. CoSMoS proposes that the domain model be expressed using a suitable software engineering notation, with labels drawn from the domain, not computation. In this way, the abstract software engineering models can be presented so that the domain expert can understand and approve (validate) the model.

The developers need to work with domain experts to find a modelling approach which supports development and is accessible and acceptable to the domain expert. It is useful to note any presumptions about the sort of simulation that will be created, and the sort of experiments that might be carried out in simulation (Droop et al. 2011; Polack et al. 2011). Revisiting and extending the assumptions collected when exploring the domain helps to situate and guide the modelling choices.

In the cell proliferation project, Petri net modelling was proposed (Droop et al. 2011; Polack et al. 2011): many biologists have seen Petri nets used to model dynamic behaviour such as signalling and pathways in biological systems (Chaouiya 2007; Glory et al. 2010; Heiner and Gilbert 2011; Materi and Wishart 2007; Ruths et al. 2008). The developers confirmed that Petri nets—which model reactive system dynamics—provide a clear and natural way to model division and differentiation. Petri net tokens represent individual cells, places represent biologically-distinguishable cell types, and transitions show the process of division and differentiation. To support seamless software engineering development, a well-defined, basic Petri net notation was selected, the HLPNG ISO standard. The notations are defined using a metamodel, which we can use to develop partly-automated transformations, ultimately to code. The development here follows a model driven engineering (MDE) approach (Czarnecki and Helsen 2006), and is covered in more detail in Polack (2012).

Several variants of the Petri net model have been produced. The first model (Droop et al. 2011) describes the four distinguishable cell types, where the basal transitions comprise three instantiations of the generic proliferation pattern shown in Fig. 4. The luminal cells do not differentiate, so have a simpler domain model (Fig. 5). Note that the generic Petri net contains all biologically possible transitions; transitions can be disabled in instantiation as needed.

The second version of the domain model comprises only basal and luminal cells. This version reflects the domain experts’ realisation that it was more efficient to count basal cells than to stain and count the three types of basal cell separately. The simplified model therefore comprises one instantiation of Fig. 4, with the start cell place being basal

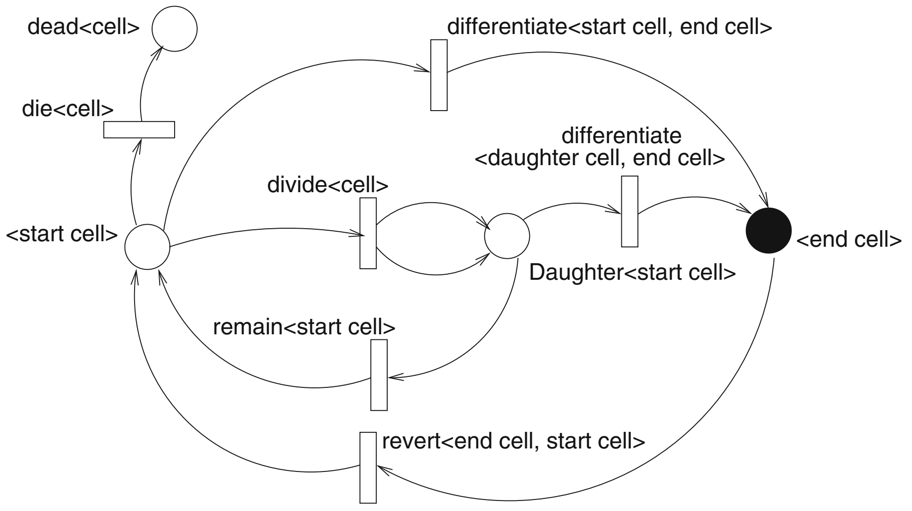

Fig. 4 Generic Petri net for cell proliferation. The full model has three instantiations of this: e.g. the first has the SC type as the start cell place and the TA type as the end cell place. *Circles* represent places; *rectangles* represent transitions; *arrows* show the direction of the movement of cell tokens. The end cell place (a *filled circle*) is a reference place: its instantiation also occurs as the start place in another Petri net instance

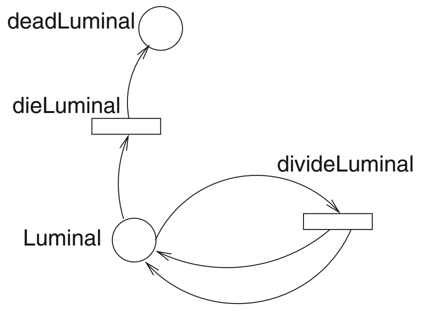

Fig. 5 Petri net for Luminal cells, which can only divide or die

and the end cell place being luminal, plus the luminal cell Petri net (Fig. 5). A further variant on this model, in which stem cells transition to basal cell, is discussed in Sect. 7.

The Petri net model provides an abstract summary of the proliferation system transitions. However, the domain experts were concerned that the model did not allow sufficient cell-level influence on the dynamics—we expect that PC and BPH arise from normal dynamics through rare cell mutations that propagate aberrant behaviour. We also need a way to introduce delays, since a cell requires energy to divide or differentiate, and has to be in an appropriate biological state. Furthermore, the Petri net semantics is that a transition consumes one token and produces another, losing the internal state of the cell. Rather than find a Petri-net variant that incorporates a mutatable internal state on tokens, and in order to maintain the clarity of distinction between cell and cell-type dynamics in the domain model, it was agreed to model the cell-level behaviours as a transformational model, using a state diagram notation (Droop et al. 2011; Polack et al. 2011). Again, a standard notation with a metamodel definition was chosen: the UML state diagram.[1] In most cases, the domain experts accept a model with two states: *active*, when a Petri net transition can be enabled; and *quiescent*, when a Petri net transition for that cell is not available—although the cell may be biologically active, its actions are not relevant to this abstraction of cell proliferation dynamics. The two-state generic state diagram is shown in Fig. 6. States and transition can be added or disabled as required.

For a seamless MDE development, the Petri net and state diagram notations need to be linked or merged. At this stage, we simply use naming conventions, and assume that each cell carries a mutable, heritable state—which the developers like to call a "pseudo-genome" (Polack 2012; Polack et al. 2011). At the domain modelling stage, a detailed consideration of cell triggering is not needed: we assume that the transitioning of cells in this combined model is controlled by a combination of transition probabilities at individual and cell-type levels (Polack 2012).

Based on the domain modelling, the collaboration can clarify the biological data (cell counts and ratios) that are required to initialise and run the simulator, as well as to calibrate the finished simulator.

5.1 Domain model assumptions

The domain model omits concepts other than those identified and agreed by the domain experts and developers, abstracting away from detail such as cell signalling, intra-cell mechanisms, biochemistry, physics, and spatial aspects. To achieve a model that is computationally feasible and simple enough to understand, the included concepts are simplified. The domain modelling assumptions relate to the way in which simplified biological concepts can be modelled, in terms of state and behaviour, but are not yet at the level of implementation decisions. Three domain model assumptions are illustrated here.

[1] See http://www.omg.org/spec/UML/.

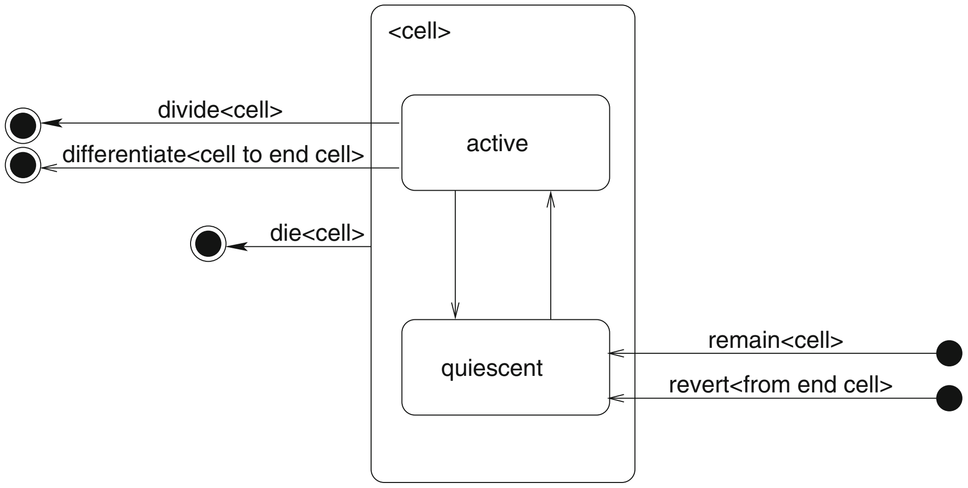

Fig. 6 Generic state diagram for individual cells. States are soft rectangles. A cell is a token in a Petri net place, so the outer state, <cell>, is instantiated with the name of the Petri net place. The names of the entry transitions (*arrows* from *filled circles*) and exit transitions (*arrows* to *circular targets*) match the labels on the Petri net transitions that produce or consume that place's tokens. A cell is produced in the quiescent state, and it can only make a proliferation transition from this Place if it is in the active state. A cell can die from any state, so the die <cell> transition originates from the outer state: it is always enabled

1. Modelling cell division as a transition from one cell to two daughter cells, each of which may then differentiate, is an acceptable way to express proliferation behaviour that results in differentiated cells. In simulation, a daughter cell must differentiate.
   - Reason: Developers proposed this simple representation of cell division. Separation of cell division and cell differentiation allows a generic representation of differentiation.
   - Justification: Domain experts observe that it is possible to identify cells that are in the process of division; modelling daughter cells potentially allows capture of simulation data on dividing cells, but also allows addition of time delays to account for the time taken to divide. The representation is also close to how the domain experts mentally visualise the process of proliferation.
2. The individual cell types can be modelled with discrete sub-states.
   - Reason: In the biological system, only some cells can divide or differentiate at any time: there is cell-level behaviour, as well as a system-level behaviour. The precise controls on division are not fully understood, but include intra-, inter- and extra-cellular factors. At the level of abstraction of our model, there needs to be an individual-cell-level aspect to determining when each cell divides or differentiates, as well as a system-level transition probability.
   - Justification: Although there are many potentially-distinguishable cell states, we are only modelling states that affect the ability to transition. Separating the state diagram of cells from the Petri net model of the system maintains clarity, whilst allowing the needed cell-level control over division and differentiation. The model allows later inclusion of cell-level mutations affecting the ability to transition.
3. The current state of a cell can be recorded in a data structure referred to as a pseudo-genome. The data provides cell-level control over division and differentiation, by being used to modify the system-level trigger condition on cell-type transitions.
   - Reason: The simulation needs to capture the variability of natural cell behaviour, so a cell-level adjustment to cell-type transition probabilities is needed. Also, a long-term goal is to simulate the effect of cell mutation on cell proliferation, and this can be achieved using a cell-level adjustment that can itself be changed.
   - Consequences: Giving each cell a mutable pseudo-genome holding values that persist (with possible mutation) over the lifetime of the cell allows rare effects to be traced through the proliferating cell population. The model also allows later inclusion of extra-cellular factors—events that cause the values stored in the pseudo-genome to change.

5.2 Domain model fitness

The foregoing discussion of the domain and domain modelling makes a range of observations on the appropriateness of modelling notations, scope and scale of modelling, and the content of the models. The CoSMoS research

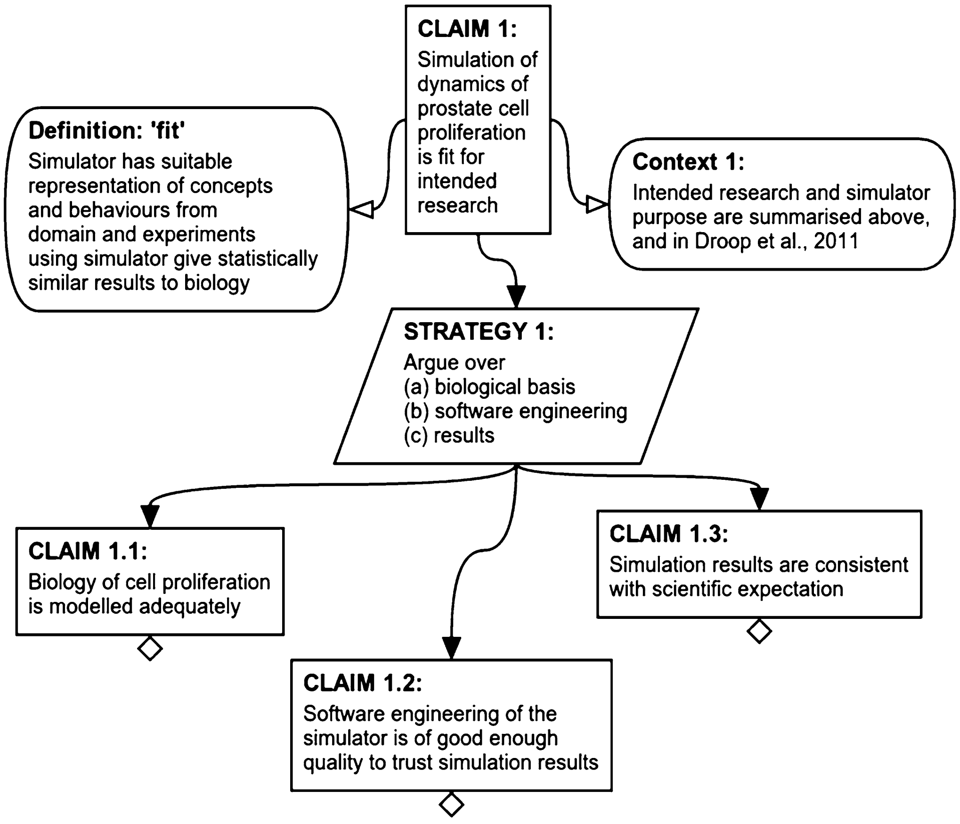
CLAIM 1:
Simulation of dynamics of prostate cell proliferation is fit for intended research
Definition: 'fit'
Simulator has suitable representation of concepts and behaviours from domain and experiments using simulator give statistically similar results to biology
Context 1:
Intended research and simulator purpose are summarised above, and in Droop et al., 2011
STRATEGY 1:
Argue over
(a) biological basis
(b) software engineering
(c) results
CLAIM 1.1:
Biology of cell proliferation is modelled adequately
CLAIM 1.3:
Simulation results are consistent with scientific expectation
CLAIM 1.2:
Software engineering of the simulator is of good enough quality to trust simulation results

Fig. 7 The structure of a top-level argument created for the fitness for purpose of the simulation. Claims are *rectangles*, with a diamond appended to indicate that the claim is not developed in this diagram. Parallelograms record strategy. Definitions and context appear in *soft rectangles*, linked using *open arrows*. All elements of the argument can link out to detailed evidence, sources, and other material from the research context

context should capture this commentary. CoSMoS work on fitness for purpose (Alden et al. 2011; Ghetiu et al. 2010; Polack 2010) also focuses on how captured rationale and agreements can be used to *argue* fitness for purpose. It is particularly important to establish the appropriateness of the domain model, because of the significant semantic gap between the biological domain and its representation as a software engineering specification. As discussed, we can reduce the risk of changing the domain expert's intention during simulator development by using well-defined software engineering notations, working with the domain expert to validate the domain model, and then by following MDE principles such as seamless development. In short, if the domain model is fit for the simulation purpose, then, by observing good software engineering practice, the implementation can also be argued to be fit for purpose. Throughout this process, we gather insights that are needed when it comes to interpreting simulation results, and subsequently if the simulator needs to be revised to address new hypotheses.

In arguing fitness for purpose, it is useful (but not essential) to capture the structure of the argument diagrammatically (Ghetiu et al. 2010; Polack 2010; Polack et al. 2011). For the cell proliferation study, we first express what it means for the whole simulator to be fit for purpose, Fig. 7, and then expand the fitness claims as the development proceeds, a structure proposed in Polack (2010). The approach addresses separately the biological basis, the software engineering and the results. The argument summary approach is based on standard safety case argumentation (Kelly 1999; Wilson and McDermid 1995) using the Goal Structuring Notation (GSN).[2] The approach is similar to evidence-based engineering proposed in critical-systems contexts (Kitchenham et al. 2004; Wu and Kelly 2007), in that initiating the whole-simulator fitness argument alongside the domain model focuses attention on the additional development obligation to demonstrate fitness for purpose, and helps in planning development and evaluation (Alden et al. 2011). Note that, in safety case argumentation, each claim is expanded, using strategies as appropriate, until *evidence* establishes each lowest level claim. However, in establishing simulator fitness for purpose, we propose that it is only necessary to fully develop every claim if the simulation results are to be used as primary evidence in establishing a biological hypothesis (Polack 2010). Here, and in most other cases, simulation results are used in a more limited way, to focus and complement laboratory research.

[2] The GSN standard is available at http://www.goalstructuringnotation.info/documents/GSN_Standard.pdf. Here, the argument structure diagrams are created using the Artoo argumentation tool, http://www.ycil.org.uk/argumentation-tool/.

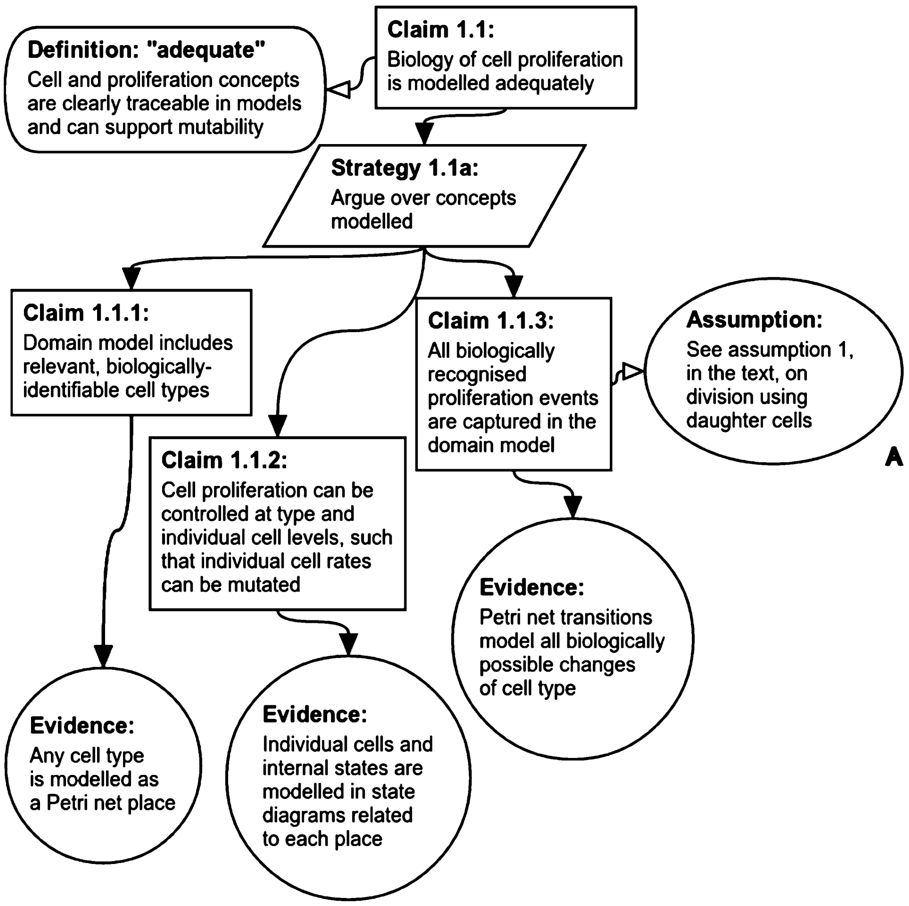
Definition: "adequate"
Cell and proliferation concepts are clearly traceable in models and can support mutability
Claim 1.1:
Biology of cell proliferation is modelled adequately
Strategy 1.1a:
Argue over concepts modelled
Claim 1.1.1:
Domain model includes relevant, biologically-identifiable cell types
Claim 1.1.2:
Cell proliferation can be controlled at type and individual cell levels, such that individual cell rates can be mutated
Claim 1.1.3:
All biologically recognised proliferation events are captured in the domain model
Assumption:
See assumption 1, in the text, on division using daughter cells
A
Evidence:
Any cell type is modelled as a Petri net place
Evidence:
Individual cells and internal states are modelled in state diagrams related to each place
Evidence:
Petri net transitions model all biologically possible changes of cell type

Fig. 8 Part of the argument expanding Claim 1.1, (Fig. 7) to consider adequate modelling of domain concepts. *Ellipses* record assumptions (and justifications); *circles* are used to summarise and point to evidence that establishes a lowest-level claim

In Fig. 7, Claim 1.1, that the biology of cell proliferation is modelled adequately, can be expanded significantly, and expansion of Claim 1.2 can be started, but Claim 1.3, concerning the results, cannot be considered until the simulator is fully developed. The fully expanded arguments cannot be reproduced here (each part is too large for the page), but two of the strategies used to expand Claim 1.1 are shown in Fig. 8—addressing the adequacy of modelling of the concepts from the domain—and Fig. 9—addressing the adequacy of scope and scale. Note that, whereas in standard GSN safety case arguments, multiple strategies are alternatives, in fitness arguments, multiple strategies are complementary.

An argument of fitness for purpose exposes our rationale to scrutiny: if a collaborator or a third party does not consider that part of the argument establishes fitness for purpose, they can challenge it, prompting revisiting or extension of the argument. To illustrate, consider Claim 1.1.5, that it is acceptable to model cell proliferation as a closed system. The evidence states that this is agreed but notes reservations. Clearly, such a statement is open to challenge: what was the basis of agreement? Why is a closed system considered to be appropriate by the domain experts? The general issue is important: open biological systems are often simulated as closed systems, but the implications of closure are not always considered. By recording this issue in the fitness argument, it is flagged to be taken into account when simulation results are considered. As a further illustration of the value of critical discussion of simulation rationale, we take the liberty of reporting one of the anonymous reviews of this paper. The reviewer makes several insightful comments, most notably that we should present an argument that the biology represented in the model is itself sufficient to explain the emergent behaviours observed in the real world.

Claim 1.2 in Fig. 7 concerns the quality of software engineering. In relation to the fitness for purpose of the domain modelling, we need to argue that the modelling notations are suitable both for the domain experts to check and for the software engineers to use for development. We have not used argumentation structures to express the software engineering aspects of fitness for purpose in this project. Instead we argue verbally, as presented above, for

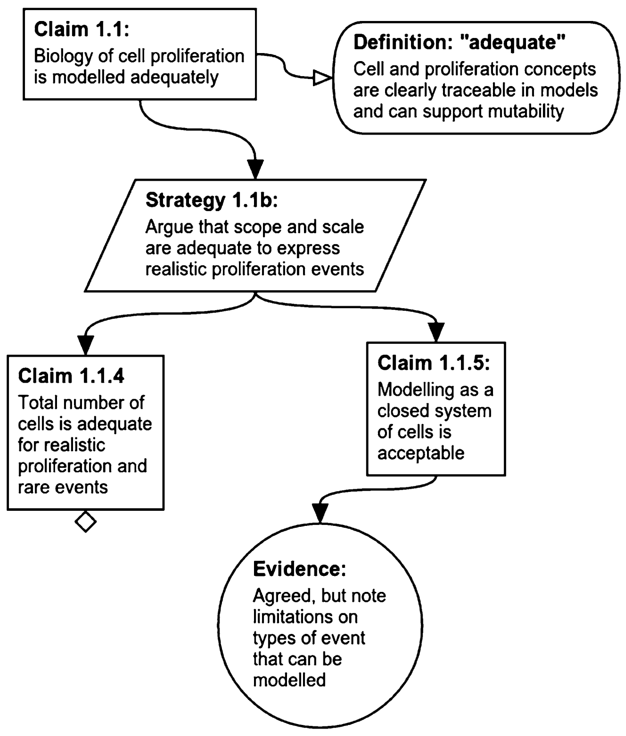
Claim 1.1:
Biology of cell proliferation is modelled adequately
Definition: "adequate"
Cell and proliferation concepts are clearly traceable in models and can support mutability
Strategy 1.1b:
Argue that scope and scale are adequate to express realistic proliferation events
Claim 1.1.4
Total number of cells is adequate for realistic proliferation and rare events
Claim 1.1.5:
Modelling as a closed system of cells is acceptable
Evidence:
Agreed, but note limitations on types of event that can be modelled

Fig. 9 Part of the argument expanding Claim 1.1 (Fig. 7) to consider adequate scope and scale. Note that the Evidence supporting Claim 1.1.5 is phrased to draw attention to the discussion of this claim in the text, rather than as strict supporting evidence

a strategy of seamless development, and we also need to demonstrate that the software has been suitably tested.

6 Platform model development

The CoSMoS platform model represents an intermediate software engineering activity that moves from the abstract biological focus of the domain model, to a concrete computational focus that leads to systematic code development that puts into practice seamless development and traceability, outlined in Sects. 2 and 4. Platform modelling must lay the groundwork for both the implementation and the evaluation of the simulator.

The effort expended on model selection and development, assumptions and argumentation needs to result in a trustworthy implemented simulator. Traceability forms part of the process of software validation (showing that the right system has been created). Software verification (showing that the system has been created correctly) needs to show that the risk of having undesired behaviour in the system is as low as reasonably practicable—the basic tenet of software testing. Code tests should be designed during platform modelling, as computational data structures and behaviours are designed.

At the system level, software engineering design normally sees planning of acceptability testing: objective criteria that the finished system must meet. However, system-level testing of simulators, and other complex systems, is not straightforward. The stochastic nature of these systems means that the system behaviour may be unexpected, and confidence in outcomes requires a statistically significant number of runs. The platform modelling can plan system level evaluation, including activities such as calibration to known domain behaviours. The platform model also contributes to evaluation by ensuring that we do not build in behaviours that should emerge from the complex interactions of lower-level components. For instance, here, the software must not pre-determine rates of cell proliferation, as these should emerge from the mechanisms triggering transitions. This is a subtle activity, and must be addressed honestly and systematically (see the work of Alden 2012; Alden et al. 2013).

To support quality development and evaluation, the platform model needs to identify a development medium that has good software engineering support, and has a clear mapping from the domain model—here, the combined Petri net and state diagram models. In the cell proliferation project, the agreed target is an agent-based simulation (Droop et al. 2011; Polack et al. 2011). However, the seamless development route and implementation language were not identified.

6.1 Seamless development options

In the domain modelling, the Petri nets and state diagrams are related by a naming convention and an assumed mechanism for co-ordinating event triggering. For platform modelling, the models need to be merged properly, checking the semantics of the mapped concepts. We then need to translate the model concepts (with biological labels) into programmatic concepts. Since both model notations have a metamodel definition in a common form, it is possible to merge the languages into a single domain-specific language (DSL) (Polack 2012). Model transformations can then be written to map from a model in the DSL, to appropriate programming languages (Paige et al. 2009). This approach supports subsequent reuse and reworking of the simulator. To create the DSL, MDE offers two options, as follows.

1. Merge the metamodels. Tool-supported MDE operations for model merging can also be applied to metamodels. In principle, metamodel merging gives the most general, semantically sound, and automatable approach. However, it is a non-trivial exercise to extract the appropriate parts of the metamodels (e.g. extracting state diagram concepts from the whole UML

Fig. 10 The schema, or metamodel, for the flattened modelling language. Places have initial cell counts. Transition rates represent the probability of the transition occurring

```
<section>    ::= "places:" (<place>)* | "trans:" (<transition>)*
<place>      ::= <place-name> <start-count>
<transition> ::= <transition-name>
   "in" <place-name>
   "out" <place-name> ("," <place-name>)*
   "rate" <number>
   ["mutability" <range>]
   (mutation)*
<mutation>   ::= "mutate" <transition-name> <range>
                 | "mutate_any" <range>
<range>      ::= <number> "to" <number>
```

```
places:              trans:               trans:
 SC_quiescent 50      aq/SC                qa/SC
 SC_active 0           in SC_quiescent      in SC_active
 dSC 0                 out SC_active        out SC_quiescent
 DeadSC 0              rate 0.01            rate 0.23
```

Fig. 11 Part of the platform model of stem (SC) and transit amplifying (TA) cell transitions, using the DSL defined in Fig. 10. Values here are arbitrary

metamodel), and to determine the generalised semantic mappings between state diagram events and Petri net transition triggering.

2. Merge the models. Using the existing name-based matches to merge the models is simpler than the metamodel merge, but at the cost of ignoring the diagrammatic concept semantics, and with the loss of the separation of levels. The solution is also model-specific, and would not generalise to other systems modelled using Petri net and state diagram notations.

Eventually, we will use the general metamodel-merging solution; meanwhile, we use a simple variant of the model-specific solution. We systematically but intuitively create a flattened textual model that contains all the information from the diagrams. The textual model follows a fixed structure with a well-defined schema (or metamodel: see Fig. 10). The process of deriving the flattened model could be defined more formally using MDE operations (Polack 2012). Figure 11 shows part of the textual model for the stem cell type and its transitions to the transit amplifying cell type, derived from the domain model instantiations of Figs. 4 and 6. Platform modelling now requires formulating implementations for the concepts represented in the flattened language.

6.2 From domain to platform concepts

The platform model is a software design, whilst the domain model from which it is derived is a model of a biological domain presented using software engineering models. Platform modelling needs to determine data, structures and behaviour from a computational standpoint and to identify the instrumentation needed to output data from the simulation.

The key design element in the platform model is the cell transition mechanism. We propose a mechanism using probabilities. Based on cell proportions in the biological data, we assign a fixed probability for cells to transition from a Petri net place. In addition, each cell's pseudo-genome records a variation from this type-level probability for that specific cell. The combined place and cell values are compared to a random variable, generated in each step of the simulation to define the transition threshold for transition. The cell-level variation not only allows for natural variability across cells; it also gives a variable to be mutated and inherited in order to simulate rare event effects implicated in the abnormal proliferation characteristics of PC and BPH.

The objective for the cell proliferation simulator is to output numbers and ratios of different cell types. For comparison with biological data, this must be done in a way that allows the expected emergent behaviour to be observed. For a simulated normal prostate, the desired emergent behaviour is stability in these values. For an abnormal prostate, the numbers and ratios of cells characteristic of PC or BPH should emerge. The simplest suitable solution is to periodically output the number of cells in each place. However, this requires a design decision on how to treat cells that are in transition at the audit point: should these be counted, or ignored? One option is to synchronise the simulator so that all transitions are complete before data is output. Another solution is to count cells in transition—a potentially useful result, since the biological samples show some cells in the process of division. The important issue is not which decision is taken, but that one option is chosen, documented in the research context, and recorded in the fitness argument for future reference.

A design decision is also required on whether to start the simulation in a mature state, or to try to generate a prostate epithelium from a few stem cells. Here, the developers and the domain expert agreed that the simulation would start with cell numbers representative of a hypothetical normal mature prostate, and with probabilities derived from observed cell ratios.

As in domain modelling, the modifications needed to turn the domain model into a software engineering design, and ultimately an implementation, that is fit for purpose, must be documented so that the models and code can be traced back to the domain model. An argument that the platform model is fit for purpose needs to establish that there is a clear, justifiable mapping from the concepts in the domain model to the concepts in the platform design; that the software development maintains the semantics and intent of the domain model; and that deletions, changes and additions are properly recorded and justified. In addition, the argument needs to pay attention to the integrity and validity (in software engineering terms) of the design. The fitness argument depends on the approach and the way that the development is carried out. As for the argument of fitness for purpose of the domain model, there is the option of creating an argument structure if this is useful.

7 To build a simulator

The simulator (simulation platform in Fig. 1), is an engineering product that implements the platform model in software. The simulator development is non-trivial, both because of the inherent complexity of the simulated processes, and because of the need to be able to convince others that the simulator is fit for purpose. As in platform modelling, the approach to simulation platform development builds directly on the design models, leading to an application in which concepts are traceable back to the domain model and the scientific domain. The implementation should be flexible enough to support a range of experiments within the stated simulation purpose. The development approach must also be feasible within the resources of the project (skills, manpower). Meeting these criteria should mean that the simulation platform is demonstrably fit for purpose.

In developing the cell proliferation simulator, we have also been exploring principles to guide selection of implementation media. We have considered concurrent object-oriented (OO), process-oriented (PO) and functional target languages. Our implementations are still at the prototype stage, so this section discuss our findings on the choice of implementation media and languages, before considering work on calibration that leads to a revisiting of the domain model and the assumptions made in this project.

7.1 A note on languages

Our goal is to be able to use MDE transformations to systematically derive code from the platform model. This approach reduces the risk of introducing errors in coding, whilst supporting traceability between model concepts and code. In addition, the transformation model can be used to derive new simulator instances from revised domain and platform models, which is important in simulator reuse and adaptation. In principle, it is easy to create a transformation model that can be used to systematically transform from a well-defined modelling language to a programming language. However, different programming languages provide different advantages and problems for transformation—in general, and in relation to different modelling contexts.

Almost all MDE work on program generation targets sequential OO languages. The assumption is that the transformation is from an OO modelling language (normally UML) that has known semantic mappings to the programming language (usually Java). However, Java does not provide a natural target for the cell proliferation platform model: whilst there is a superficial resemblance of places and cell tokens to classes and objects, the semantics are rather different (Polack 2014), and our platform model semantics is simpler than the OO message-passing model.

Agent models have been shown to map cleanly to PO languages (Polack et al. 2005; Ritson and Welch 2010) that support mobility and concurrency, such as occam-π and, more recently, Go. These PO languages are based on the formalism of communicating sequential processes, and there are some guidelines on programming styles that promote clean concurrent programs, and even proof of healthiness properties if required (Sampson 2010; Welch and Barnes 2005; Welch and Barnes 2008). To map the cell proliferation domain model to PO code requires mapping the cells and cell types to processes and process types; transitions can be modelled by passing mobile cell processes across a static network of cell-type processes (Polack et al. 2005; Ritson and Welch 2010).

Although PO seems a natural choice for the cell proliferation simulation platform, in software engineering terms, PO development is compromised by relatively poor tool support (compared to OO languages such as Java). Go is too new to have attracted significant publicly-available software engineering support, whilst occam-π is primarily an academic language. Our first attempt at implementing the simulator tried to circumvent this problem by using JCSP, a PO library for Java. JCSP was developed by the group responsible for occam-π, and observes the formalism of communicating sequential processes (Polack et al. 2011). Being Java-based, it can exploit Java programming environments. However, designing for a hybrid OO-PO language is non-trivial, and the Java programming environments do not extend support to the PO concepts in JCSP. Furthermore, the Java-based concepts required a partial OO mapping, obviating the advantages of the agent-to-process mapping. The most recent PO prototype is using Go,[3] an open-source language that builds PO structures on a Python base. This benefits from the simplicity and development support for Python.

The simplest transformation to code that we have identified is from the textual platform model to the concurrent functional language, Erlang. The MDE transformation model to Erlang has been partially automated using the Epsilon Generation Language (Kolovos et al. 2011; Rose et al. 2008). The generated code is readable and traceable. Because code is generated, verification (testing) is applied to the transformations: if correct transformations are applied to correct models, then code is correct by construction. However, this approach has also proved problematic. The Erlang language, like most functional languages, has no inherent state, and also introduces challenges for both execution and calibration. For instance, Erlang uses lazy evaluation, so has to be forced to generate a new random number for each probability evaluation. The most significant problem with the Erlang implementation is that transition probabilities have to be hard-coded; this is discussed further in the next section.

7.2 Calibration and its consequences

The simulation platform and the models from which it is developed are abstractions of biological domain concepts. Calibration is part of the process of testing the simulator, but is rarely used in areas other than simulation. Read (2011), in seminal work on calibration in the CoSMoS context, describes three reasons for using calibration on simulations of complex systems.

1. When designing and developing the simulator, there is no way to tell whether the abstractions and assumptions will adequately reproduce the intended real-world behaviours.
2. Biological values required for the simulation may be unknown.
3. Known biological values may not map directly to simulation parameters.

These three points apply to the cell proliferation study, in which cell counts need to be converted into transition probabilities to create scenarios that exhibit realistic characteristics; there is limited biological data; and there is wide variation across biological samples (Sect. 4).

Calibration needs to consider parameter values individually and in combination, and may reveal weaknesses in the implemented model. The successful parameter sets discovered by calibration are computational solutions: they need to be reviewed by the domain experts, to reject biologically-implausible parametrisation.

Even for the cell proliferation model, the scale of parameter uncertainty makes calibration a challenging activity. A student project has started exploring its calibration. A prototype calibration tool has been built for the Erlang simulator, that automatically creates and evaluates sets of parameter values. Because Erlang parameters are hard-coded (Sect. 7.1), the tool includes a wrapper to insert the parameters directly into the code. Calibration has not yet been applied to the Go simulator.

The calibration project compares random hill climbing and simple evolutionary algorithms, and shows that random hill climbing outperforms evolutionary algorithms on this task. In future, we will consider other search algorithms, such as multi-objective meta-heuristic search, which has recently been applied successfully to simulator calibration (Read et al. 2013).

The calibration attempts to mimic what is known of epithelial cell dynamics in a normal prostate, defined by the domain experts as maintaining a roughly-constant number of cells and a ratio of basal to luminal cells that is always between 2:3 and 3:2. This is translated to a fitness function that rewards parameter sets that give long-term cell ratios characteristic of the normal prostate. There is scope for improving the fitness function. However, a key early insight from calibration is that, in a system with so much variation in the biological samples, calibration can produce many candidate parameter sets that give behaviour that could be said to characterise normal prostate behaviour: we do not just produce a single calibrated model.

Calibration does more than just find parameters; it can also reveal issues related to domain modelling and domain understanding (Read et al. 2012; Read 2011). In initial runs of the cell proliferation calibration, all cells died out. This behaviour persisted even with unrealistically-small cell-death probabilities. Further analysis identified an over-simplification of the domain model. In generalising stem, transit amplifying and committed basal cells to just basal cells (Sect. 5), we had removed the important ability of the cell population to regenerate from a small number of practically-immortal stem cells. The solution required re-introducing a small number of stem cells (approximately 1% of cells, based on estimates by domain experts). The change is propagated through the CoSMoS products: the domain model is extended with a Petri net representation of stem cells that can only divide to give new basal cells; the

[3] http://golang.org/.

platform model is extended with the new cell type and its basic division behaviour, and thus into the simulator Erlang code. Rerunning the revised simulator shows that simulation runs rarely terminate with total cell death.

In analysing the die-out problem, we identified a more significant problem. The Erlang simulator uses fixed transition probabilities for each run, so it cannot adjust division rates in response total cell numbers. The model thus ignores an important domain observation, that normal cell proliferation is inhibited by crowding. Some other issues that were identified as a result of calibration activity are as follows.

- How long do we need to run a simulation? Can we estimate an upper limit for the number of successive cell transitions for a prostate epithelial cell?
  - A cell takes about 20 h to divide; during this time it replicates all its proteins and DNA. This takes energy, so the cell cannot divide again until its energy resource recovers. A mature prostate might last for 70 years. From this, the domain experts make a worst-case estimate that the absolute upper limit for the number of times that one cell (e.g. an embryonic stem cell) could divide in a lifetime is in the region of two thousand. Calibration has identified parameter sets that produce stable behaviour over several thousand transitions (though most then show a falling total cell population). However, the stable sequences are not usually at the start of simulation—it is unclear why there is a transient period before stability.
- How many simulated cells are needed to give stable behaviour?
  - Calibration shows that simulation runs with some parameter sets take longer to stabilise than others. Furthermore, runs with large starting populations stabilise faster than runs with smaller populations. We cannot observe living prostate cell proliferation dynamics—that is why we are creating a simulator—but the domain experts can estimate the typical number of cells in the prostate epithelium. Our original intention was to create a simulator for a whole prostate epithelium, comprising billions of cells. However, it has become obvious from looking at the biopsy samples being analysed in the laboratory that much of a prostate may appear normal even when small parts clearly show PC or BPH cell characteristics, and that our simulator would need to focus on a small section of prostate epithelium if we were to be able to reliably show the emergence of PC or BPH cell characteristics. Initial calibration finds that stable, normal behaviour on populations arises in most runs of 100,000 cells (with runs starting from 60,000 luminal and 40,000 basal cells), but that with a population of 10,000 cells (same initial ratio) most runs did not produce stable behaviour. This finding needs further consideration at both domain and simulation platform levels.
- Following from discussion of cell numbers, is the decision to create a non-spatial simulation sustainable?
  - The original decision to develop a non-spatial simulation was based on the assumptions (by the developers) that there would be enough cell-count data to characterise a normal prostate, PC and BPH, and (by the domain experts) that a student intern could produce these cell counts, given suitable materials. Over the following three years, a range of difficulties with sample quality, staining protocols, and counting technology has shown (a) that although we have more cell count data than most published research on the prostate, we do not have enough to provide generalised aspatial data to the simulator—but we do have a good set of sample images against which we could compare snapshots of simulated spatial behaviour; and (b) that the spatial layout of cells, particularly of dividing cells in normal and non-normal samples, is interesting. In known BPH samples showing BPH cell characteristics, dividing basal cells are clustered, whereas in samples with normal cell characteristics, division is rare and isolated. It is thus possible that a 2D visualisation that represents the epithelial layer of a section of prostate, mimicking the tissue slices being used for cell counting, would be easier to calibrate and interpret than cell counts and ratios alone.
- What is the probability of a cell dying from any state?
  - As in the domain model, the simulator provides transitions to dead for all types of cell, allowing reporting of the number of simulated cells of each type that have died at any point. However, we cannot get matching biological data, as dead cells are broken down into indistinguishable chemical components. In theory, dying cells can be counted using stains that bind to chemicals expressed by dying cells, but laboratory tests find that the stains also bind to many cells damaged in slicing—and the stain does not discriminate the type of the dying cell. We conclude that the probability of dying must be determined through calibration, with reference to domain experts to check that our eventual death transition probabilities are realistic.

7.3 Next steps

The process of developing and calibrating the prototype cell proliferation simulator has raised many new questions about the domain, and about assumptions and design decisions taken. Continual iteration is a feature of most CoSMoS-related projects working on collaborative research simulators (Greaves et al. 2012, 2013; Read et al. 2012, 2013; Read 2011; Williams et al. 2013).

It is now clear that the next iteration of the cell proliferation simulation project needs to consider a spatial model, and to focus on a characteristic section of epithelium that can produce "slices" to compare to the biopsy samples used in the laboratory research. As well as facilitating calibration, a spatial model will allow better modelling of the feedback between cell death and cell division: cell proliferation could be automatically adjusted according to a measure of spatial crowding.

The issues raised by the initial calibration work on the simulator reflect back on the fitness for purpose of the biological model. In the fitness for purpose argument in Fig. 9 (Sect. 5.2) there is now a clear omission from the claims proposed under the strategy of arguing adequacy of the scope and scale of the simulator: we also need to address the initial decisions to create an aspatial model of the whole prostate epithelium, in the light of our improved understanding of the simulator, the biological domain and the cell data that is available to the project.

8 Discussion and Conclusion

This paper presents the principled development of a cell proliferation simulator, comprising simple interactions at system and cell levels. The project involves collaboration between software engineers and domain expert biologists, both in modelling and design and in validation and calibration. The project illustrates the use of the CoSMoS approach, highlighting the phases and products. A notable aspect of this project is that it has never had dedicated funding, but by abiding by CoSMoS principles, we have been able to induct a series of student and interns to progress the modelling, implementation, calibration and biological cell counting aspects of the research. The project also contributes to the evolution of the CoSMoS approach, showing how models at different levels of abstraction can be used, and outlining the model-driven engineering approaches that can improve the traceability of domain to code.

Simulation is used because cell proliferation dynamics cannot easily be studied *in* or *ex vivo*. Agent based models are used because we wish to be able to introduce variation and mutation at the individual cell level, and to understand the effects of rare mutation on populations of cells. We use the CoSMoS approach and MDE principles because, although we are producing a simulator that is as simple as we can make it, we need a product that is flexible, and can be reliably reworked and reused to support the ongoing biological research into prostate conditions in which cell proliferation is implicated.

The initial project purpose remains to be completed. As well as the issues and challenges noted in this article, many new challenges are appearing as the biologists begin to think in terms of being able to experiment with possible dynamic proliferation behaviours. Even without a full working research simulator, the collaboration has produced many valuable side effects for both sides.

Because the simulator is stochastic (probabilistic transitions), it is not sufficient to present the results of a single run. Any simulation experiment must be run many times. Although the development has created concurrent implementations of the simulator, each computational experiment takes significant time, with the runs needed for calibration or exploration taking many hours. We have not yet set up the scripting to run many experiments at once on a grid or cluster, but this is clearly a necessity for the exploratory phase. In addition, we need to identify the required number of runs and appropriate statistical analyses needed to compare simulator and laboratory results. Fitness for purpose in relation to the results might require consideration of the appropriateness of the number of runs used; appropriateness of statistical descriptors (i.e. selection of measures of central tendency, dispersion, and skew); and appropriateness of statistical comparisons (confidence, etc.)

We have shown above how the development of the simulator and the calibration form part of an argument (which may be expressed formally or informally) that the simulator is fit for purpose. It is often the case that a scientist faced with a computer program assumes that this magical apparatus is correct and gives meaningful results. Whilst the argument of fitness for purpose naturally focuses on capturing the reasoning behind the developers' belief in the simulator quality, it is equally important that the argument can convey to future audiences an understanding of the limitations of software engineering.

In analysing calibration activities, we describe how discoveries made late in development can cause not only models and design decisions but also fitness for purpose to be revisited. The fitness for purpose argument determines the conditions of use for the simulator, and simulation results should be interpreted only within the context of the fitness for purpose. Calibration has shown that the simulator can reproduce apparently normal cell proliferation behaviour.

Superficially, the interpretation of the calibration (and ultimately the simulation experiment) results is straightforward. However, the actual research context is more complicated, because the biological observations are too sparse and varied to provide good calibration and comparison contexts. Although laboratory research stimulated by the project has stained and counted significantly more prostate cells than any published study, there are no time series data (sequences of samples from the same organ), and no way to determine whether each sample represents normal, PC, BPH or some other prostate condition. Ongoing work on calibration, and future work refocusing the simulator, continue to follow the CoSMoS approach, involving both biological domain experts and developers.

Acknowledgments Fiona Polack's contribution includes work undertaken for the EPSRC CoSMoS project, grants EP/E053505/1 and EP/E049419/1. Alastair Droop's contribution has been funded through the Maitland Lab's Yorkshire Cancer Research platform grant, and from the EPSRC TRANSIT "bridging the gap" funding through the York Centre for Complex Systems Analysis (grant EP/F032749/1). We would like to thank Professor Norman Maitland and the staff of the Maitland Lab, Department of Biology, University of York, for their ongoing involvement in this project: the research underlying this paper is funded by a range of organisations notably Yorkshire Cancer Research. We also acknowledge significant contributions to the biological domain from Biology interns Sarah Greener and Emily Pollard; and to the simulation development and theory by Computer Science undergraduate project students, Jon Low, Nikola Nikolov and Codrin Oprea; and YCCSA Summerschool students Livia Dia (2013) and Andrei Simionescu (2012). The Wellcome Trust funded biological materials and the 2012 summer internship through York's C2D2 grant.

References

Albergante L, Timmis J, Beattie L, Kaye PM (2013) A Petri net model of granulomatous inflammation: Implications for IL-10 mediated control of Leishmania donovani infection. PLoS Comput Biol 9(11):e1003334

Alden K (2012) Simulation and statistical techniques to explore lymphoid tissue organogenesis. PhD thesis, University of York, New York. http://etheses.whiterose.ac.uk/3220/

Alden K, Andrews P, Timmis J, Veiga-Fernandes H, Coles MC (2011) Towards argument-driven validation of an in-silico model of immune tissue organogenesis, vol 6825 of LNCS. In: ICARIS, Springer, New York, pp 66–70

Alden K, Read M, Timmis J, Andrews P, Veiga-Frenandes H, Coles M (2013) Spartan: a comprehensive tool for understanding uncertainty in simulations of biological systems. PLoS Comput Biol 9(2):e1002916

Alden K, Timmis J, Andrews PS, Veiga-Fernandes H, Coles MC (2012) Pairing experimentation and computational modelling to understand the role of tissue inducer cells in the development of lymphoid organs. Front Immunol 3:172

Andrews PS, Polack FAC, Sampson AT, Stepney S, Timmis J (2010) The CoSMoS Process, version 0.1. Technical Report YCS-2010-450, Department of Computer Science, University of York. www.cs.york.ac.uk/ftpdir/reports/2010/YCS/453/YCS-2010-453.pdf.

Andrews PS, Stepney S, Hoverd T, Polack FAC, Sampson AT, Timmis J (2011) CoSMoS process, models and metamodels. In: Workshop on complex systems modelling and simulation. Luniver Press, Oxford, pp 1–14

Claudine C (2007) Petri net modelling of biological networks. Brief Bioinform 8(4):210–219

Czarnecki K, Helsen S (2006) Feature-based survey of model transformation approaches. IBM Syst J 45(3):621–645. doi:10.1147/sj.453.0621

Droop A, Garnett P, Polack FAC, Stepney S (2011) Multiple model simulation: modelling cell division and differentiation in the prostate. In: Workshop on complex systems modelling and simulation. Luniver Press, Oxford, pp 79–112

Forrester JM, Greaves RB, Polack FAC (2012) CoSMoS in the context of social ecological research. In: Workshop on complex systems modelling and simulation. Luniver Press, Oxford, pp 47–76

Garnett P, Steinacher A, Stepney S, Clayton R, Leyser O (2010a) Computer simulation: the imaginary friend of auxin transport biology. BioEssays 32(9):828–835

Garnett P, Stepney S, Day F, Leyser O (2010b) Using the CoSMoS process to enhance an executable model of auxin transport canalisation. In: Workshop on complex systems modelling and simulation. Luniver Press, Oxford, pp 9–32

Garnett P, Stepney S, Leyser O (2008) Towards an executable model of auxin transport canalisation. In: Workshop on complex systems modelling and simulation. Luniver Press, Oxford, pp 63–92

Ghetiu T, Polack FAC, Bown J (2010) Argument-driven validation of computer simulations: a necessity rather than an option. In: VALID, IEEE, pp 1–4

Glory NGD, Emerald JD (2010) Petri net models and non linear genetic diseases. In: IEE bio-inspired computing: theories and applications, pp 1466–1470

Greaves RB, Read M, Timmis J, Andrews PS, Butler JA, Gerckens B, Kumar V (2013) In silico investigation of novel biological pathways: the role of CD200 in regulation of T cell priming in experimental autoimmune encephalomyelitis. Biosystems. doi:10.1016/j.biosystems.2013.03.007

Greaves RB, Read M, Timmis J, Andrews PS, Kumar V (2012) Extending an established simulation: exploration of the possible effects using a case study in experimental autoimmune encephalomyelitis. In: Information processing in cells and tissues, vol 7223 of LNCS. Springer, New York, pp 150–161

Heiner M, Gilbert D (2011) How might Petri nets enhance your systems biology toolkit. In: Petri Nets, vol 6709 of LNCS, Springer, New York, pp 17–37

Kelly TP (1999) Arguing safety: a systematic approach to managing safety cases. PhD thesis, Department of Computer Science, University of York, YCST 99/05.

Kitchenham BA, Dyba T, Jorgensen M (2004) Evidence-based software engineering. In: IEEE Computer Society, ICSE, pp 273–281.

Kolovos D, Rose L, Garca-Domnguez A, Paige R (2011) The epsilon book. Online. http://www.eclipse.org/epsilon/doc/book/

Louis MR, Paige RF, Kolovos DS, Polack FAC (2008) The epsilon generation language. In: Model driven architecture foundations and applications, vol 5095 of LNCS. Springer, New York, pp 1–16

Maitland NJ, Collins AT (2008) Prostate cancer stem cells: a new target for therapy. J Clin Oncol 26(17):2862–2870

Maitland NJ, Tindall DJ (eds) (2013) Prostate cancer: biochemistry, molecular biology and genetics, chapter stem cells in the normal and malignant prostate. Number 16 in Protein Reviews. Mayo Clinic. doi:10.10007/978-1-4614-6828-8_1

Materi W, Wishart DS (2007) Computational systems biology in cancer: modeling methods and applications. Gene Regul Syst Biol 1:91–110

Moore J, Moyo D, Beattie L, Andrews P, Timmis J, Kaye P (2013) Functional complexity of the Leishmania granuloma and the potential of in silico modelling. Front Immun 4(35). doi:10.3389/fimmu.2013.00035

Paige RF, Kolovos DS, Rose LM, Drivalos N, Polack FAC (2009) The design of a conceptual framework and technical infrastructure for model management language engineering. In: IEEE Computer Society, ICECCS, pp 162–171

Polack FAC (2010) Arguing validation of simulations in science. In: Workshop on complex systems modelling and simulation. Luniver Press, Oxford, pp 51–74

Polack FAC (2012) Choosing and adapting design notations in the principled development of complex systems simulations for research. In: Modelling the physical world at models. ACM Digitial Library

Polack F (2014) Filling gaps in simulation of complex systems: the background and motivation for CoSMoS. Natural Computing. May 2014.

Polack F, Stepney S, Turner H, Welch P, Barnes F (2005) An architecture for modelling emergence in CA-like systems. In: European conference: advances in artificial life, vol 3630 of LNAI. Springer, New York, pp 433–442

Polack FAC, Hoverd T, Sampson AT, Stepney S, Timmis J (2008) Complex systems models: engineering simulations. In: ALife XI, MIT Press, Cambridge, pp 482–489

Polack FAC, Andrews PS, Sampson AT (2009) The engineering of concurrent simulations of complex systems. In: CEC, IEEE Press, pp 217–224

Polack FAC, Andrews PS, Ghetiu T, Read M, Stepney S, Timmis J, Sampson AT (2010) Reflections on the simulation of complex systems for science. In: ICECCS, IEEE Press, pp 276–285

Polack FAC, Droop A, Garnett P, Ghetiu T, Stepney S (2011) Simulation validation: exploring the suitability of a simulation of cell division and differentiation in the prostate. In: Workshop on complex systems modelling and simulation. Luniver Press, Oxford, pp 113–133

Read MN (2011) Statistical and modelling techniques to build confidence in the investigation of immunology through agent-based aimulation. PhD thesis, University of York, York

Read M, Andrews PS, Timmis J, Kumar V (2009a) A domain model of experimental autoimmune encephalomyelitis. In: Workshop on complex systems modelling and simulation. Luniver Press, Oxford, pp 9–44

Read M, Andrews PS, Timmis J, Kumar V (2009b) Using UML to model EAE and its regulatory network. In: ICARIS, vol 5666 of LNCS. Springer, New York

Read M, Andrews PS, Timmis J, Kumar V (2012) Techniques for grounding agent-based simulations in the real domain: a case study in experimental autoimmune encephalomyelitis. Math Comput Model Dyn Syst 18(1):67–86

Read M, Tripp M, Leonova H, Rose L, Timmis J (2013) Automated calibration of agent-based immunological simulations. In: ECAL, MIT Press, Cambridge, pp 874–875. doi:10.7551/978-0-262-31709-2-ch129

Ritson CG, Welch Peter H (2010) A process-oriented architecture for complex system modelling. Concurr Comput 22:182–196

Ruths D, Muller M, Tseng JT, Nakhleh L, Ram PT (2008) The signaling Petri net-based simulator: a non-parametric strategy for characterizing the dynamics of cell-specific signaling networks. PLoS Comput Biol 4(3):e1000005

Sampson AT (2010) Process-oriented patterns for concurrent software engineering. PhD thesis, University of Kent, Canterbury

Software and systems engineering: High-level Petri Nets Part 2: Transfer format

Stepney S (2012) A pattern language for scientific simulations. In: Workshop on complex systems modelling and aimulation. Luniver Press, Oxford, pp 77–103

Welch PH, Barnes FRM (2005) Communicating mobile processes: introducing occam-pi. In: 25 Years of CSP, vol 3525 of LNCS, Springer, New York, pp 175–210

Welch PH, Barnes Frederick RM (2008) A CSP model for mobile channels. Communicating process architectures 2008. IOS Press, WoTUG, pp 17–33

Williams RA, Greaves R, Read M, Andrews PS, Kumar V (2013) In silico investigation into dendritic cell regulation of CD8Treg mediated killing of Th1 cells in murine experimental autoimmune encephalomyelitis. BMC Bioinform 14(Suppl 6):S9

Wilson SP, McDermid JA (1995) Integrated analysis of complex safety critical systems. Comput J 38(10):765–776

Wu W, Kelly T (2007) Towards evidence-based architectural design for safety-critical software applications. In: Architecting dependable systems, vol 4615 of LNCS. Springer, New York