Simulation validation: exploring the suitability of a simulation of cell division and differentiation in the prostate

Fiona A. C. Polack, Alastair Droop, Philip Garnett, Teodor Ghetiu, and Susan Stepney

YCCSA, University of York, UK, YO10 5DD
Fiona.Polack@cs.york.ac.uk

Abstract. Individual or agent-based simulation is an important tool for research involving understanding of complex systems. For a research tool to be useful, its use must be understood, and it must be possible to interpret the results of using the tool in the context of the research. This paper presents the partial validity argument for ongoing work on prostate cell simulation (a companion paper describes the models and implementation of the simulation). This is the basis for a discussion of issues in the validation of complex systems simulations used as scientific research tools.
Keywords: Complex systems, agent based simulation, validity, argumentation, prostate

1 Introduction

The use of individual-based or agent-based simulation as a scientific research tool requires both good software engineering and robust modelling. For a research tool to be useful, its use must be understood: it must be possible to interpret results from the tool in the context of the research (see, for example, [12, 18, 22]).

This paper presents the partial validity argument for ongoing work on prostate cell simulation, and uses this example as the basis for discussion of issues in the validation of complex systems simulations used as scientific research tool. The paper complements Droop et al's description of the modelling and implementation of a simulation of prostate cell division and differentiation [7].

This section briefly summarises the background to the prostate cell simulator, its modelling and validation. Section 2 introduces a partial validity argument for the cell division and differentiation model. In the discussion (Section 3), we consider issues identified in the development, and subsequent review, of the validity argument. Section 4 presents some of the further work that is needed on the prostate cell simulator and to support use of complex systems simulation as a scientific instrument.

1.1 Modelling prostate cell division and differentiation

The companion paper [7] establishes the biological basis for the simulation of prostate cell division and differentiation, which is the first phase in develop-

ment of a simulator that will support the study of cancer neogenesis. The first-phase simulator will replicate observed cell population dynamics in a "normal" prostate: calibration is against laboratory data on prostate cell numbers and proportions. The full project will explore cancer as an emergent result of rare-event mutations and cell division and differentiation; thus the first phase has to develop a simulator that allows later addition of mutability and heritability.

1.2 Development of the prostate cell simulator

The prostate cell model simulator [7] is developed following the CoSMoS process, a principled approach to modelling and simulation [3, 16]. In CoSMoS, a simulator is a platform on which simulations are run: the simulator is built to support a specific area of research, or *purpose*. The CoSMoS process starts by identification of a domain of interest and of the domain expert(s) who are the primary source of domain information and understanding. A problem faced by many simulation developers is that there is disagreement among experts as to the behaviours, or even structures, of a particular subject; working in isolation, a developer has to try to extract a coherent view of the domain. A fundamental aspect of the CoSMoS process is that development takes place in collaboration with an explicit, specific group of domain experts. The simulator is designed to express the domain experts' understanding.

Having established the domain experts, the domain model is produced, in close collaboration between developers and domain experts. Early and continuous involvement of domain experts helps to define the purpose, scope and scale of the simulation exercise, the *research context*. From the domain model, developers derive a platform model, a conventional software (or hardware) design. The research context is elaborated with modelling and design decisions, simplifications and assumptions. A simulator is built from the platform model, and is subject to both testing (to establish the quality of the implementation) and calibration (to establish the accuracy of parameters, behaviours etc. using simulation runs initialised to known biological parameters). Throughout the development of the simulator, the research context can be supplemented with a record of sources, assumptions, design decisions, interpretations, etc [3, 16]. The CoSMoS process does not end with implementation. Simulations are run on the simulator, to test or develop hypotheses of relevance to the domain. It is necessary to interpret data from a simulation, or observations made in watching a visual simulation, into the domain of research: a simulator is not an exact replica of reality, and data collected from a simulation is about the agents in the simulation, not about concepts in reality. The research context is used to understand the mappings between real and simulated concepts, and the limitations of the simulation.

The domain for the project described here is the prostate cell division and differentiation model summarised in [7]. The domain expert for this simulation exercise is a group of researchers from Maitland's Yorkshire Cancer Research lab at the University of York[1]. The development team (modellers and coders)

[1] www.york.ac.uk/biology/units/cru/

comprises the authors of [7]. Droop is a developer and a domain expert; his roles are: to identify biological issues as they arise; to provide background and interpretation of the biology for the developers; and to set up review meetings at which both developers and lab members are represented.

Of the following three specific goals for the research, this paper relates only to the simulation for the first goal; the later goals motivate the purpose of the first simulation.

1. Develop a simulation of the Maitland Lab's cell differentiation and division model, based on prostate cell populations from laboratory research. The *purpose* of the model is to replicate observed cell population dynamics, represented as changing proportions of cells in a "normal" prostate.
2. Building on the model of the "normal" prostate, work with the Maitland Lab to develop simulations that capture known environmental variation and mutation. The purpose of the model is to explore the emergence of cell proportions indicative of cancer (or other prostate conditions).
3. Using these models of normal and cancerous prostate cell behaviours, develop simulation experiments that can be used to guide and test laboratory hypotheses of cancer development and control.

Understanding the purpose of the simulation and the uses to be made of it focuses the scope, scale and level of the simulation. To create the model of cell differentiation and division, the developers need to understand the cell biology at an appropriate level, and to be able to engineer environmental interaction. To meet the research purpose, it must be possible to monitor the proportions of different cells in the simulated prostate.

The domain and platform models used in developing the first-phase prototype implementation of the cell division and differentiation model. The domain model comprises two levels. The high-level model of cell division and differentiation uses a Petri net (with novel firing semantics). We model the cell-level behaviours with (a) a state diagram for the behaviours of cells in each place and transition in the Petri net; (b) a class diagram (with agent, rather than object semantics) for structure; (c) sequence charts to show how cells are consumed and produced by transitions. The domain model was developed iteratively, guided by the domain experts. From the domain model, a platform model was developed: the structure and design decisions are summarised in [7], which also describes the first prototype agent-based simulator, implemented in JCSP. Unusually for a simulation of a complex biological system, there is a clean mapping from concepts in the domain through to implementation.

In producing the first prototype simulator, there were four major review meetings, at which biological understanding and detail of diagrammatic models were discussed and revised, until all were confident that the domain models adequately represent the biological understanding of the laboratory researchers. The records of the meetings provide much of the evidence needed to establish the biological validity (or fitness for purpose) of the simulator.

1.3 Validity Argumentation

As part of CoSMoS, we have been investigating validation of simulations (e.g. [8, 9, 15, 16]). Building on traditional simulation development (e.g. [20]) and work in critical systems engineering (e.g. [1, 13]), we propose a case for the validity, or fitness-for-purpose, of a simulation as a structured argument over evidence. This section reviews our existing work on validity arguments, and introduces a notation for summarising arguments. Section 2 then presents a partial validity argument for the cell division and differentiation model.

Validation of a complex system simulation can never be absolute; it can, at best, express the basis on which we believe that the simulation is fit for its intended purpose. A key observation is that the validity argument may be incomplete without losing its value. Unless a simulation is used as primary evidence in research and research publications, a thorough and properly documented validation exercise may be unnecessary. Polack [15] notes the importance of the validation exercise itself, and the mindset that goes with the focus on capturing validity arguments and evidence, in generating a strong collaboration and in raising the profile of simulation as a tool in scientific research.

When constructing and presenting an argument, it is useful to provide a diagrammatic summary. This exposes the structure of the argument so that it can be understood and reviewed. There are many possible notations for expressing the structure of an argument: we use the Goal Structuring Notation (GSN) [13, 23], a notation devised to support safety case arguments. A GSN diagram shows a hierarchy from the top-level claim, through sub-claims that support that claim, and eventually to the evidence supporting the claims. The core notation is summarised in Figure 1.

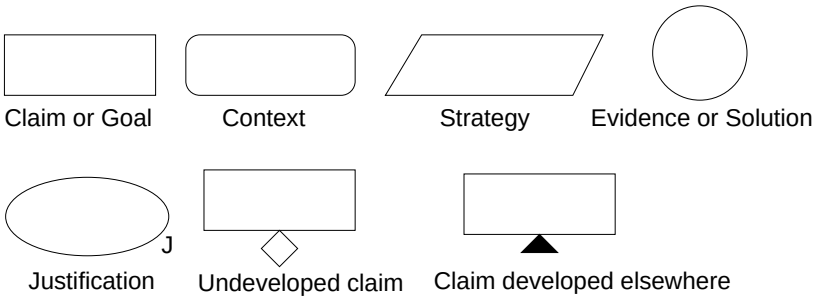

Fig. 1. Basic GSN notations [13, 23]. In safety case arguments, undeveloped *goals* are always subsequently expanded to *solutions*. This is not the case in validity arguments, where claims may be left undeveloped; in validity arguments, end point is a reference to *evidence* to substantiate the claim.

The GSN diagram is only a summary of an argument; it is intended to provide an overview or index to supporting material. Notations such as GSN also support

expression of generic arguments and argument patterns [8, 21]. Here we use some patterns for simulation validity argument that we have identified elsewhere [8, 9, 15]. As in Weaver's safety case argument patterns [21], however, we find that the use of patterns can reduce the insight gained during argument construction. Pattern use is most appropriate for high-level structuring and for systematic claims such as those relating to the statistical analysis of results.

1.4 Validity arguments vs safety case arguments

In safety case argumentation, it is the argumentation culture and the safety-case literature that make safety case argumentation a powerful tool. In using argumentation in the validation of simulations for research purposes, we similarly aim to influence the culture of development and research. However, there are some differences between safety case argumentation and our validity argumentation.

In safety critical systems engineering, a GSN argument is used to present a safety case [13]. The safety case must be a complete argument, in which evidence is provided in support of all the claims. Recent work has focused on completeness and clarity of safety arguments, and the need for side-arguments to cover the motivation and justification of the safety case (e.g. [11]). A validity argument is likely to present a much less rigorous case. We do not commit to producing complete arguments, being concerned primarily with capturing the rationale for a shared belief in fitness for purpose.

Safety case arguments are constructed to be reviewed by independent authorities, which determine what evidence is and is not acceptable. By contrast, a validity argument does not usually have a regulator to judge the acceptability of evidence, and is not automatically exposed to wider review (though wider exposure is important if the subject of validation is a high-impact or critical research simulation); the argument instead captures the mutual understanding of domain experts and developers. Typically, a validity argument must satisfy both parties, and acceptable claims, strategies and evidence are those that demonstrate to the participants that the simulation is fit for purpose.

2 The Prostate Model Validity Argument

The validity argument in this section was constructed during the development of the cell simulator: the cell division and differentiation domain model was complete, the platform model was well advanced, and an initial prototype simulation was under development. It is thus necessarily incomplete (e.g. we had no results).

The argument is based on a top-level claim that the simulation is fit for purpose, Figure 2. Context 1 points to where the intended purpose is explained. There are many ways to demonstrate the fitness-for-purpose of a simulation: Strategy 1, based on a pattern we have used elsewhere [15], addresses separately the biological basis, the software engineering and the results.

The rest of this section presents an expansion of two of the three sub-claims: Claim 1.1, concerning the adequacy of the modelling of the prostate cell behaviour, and Claim 1.2, concerning software engineering quality. Issues relating

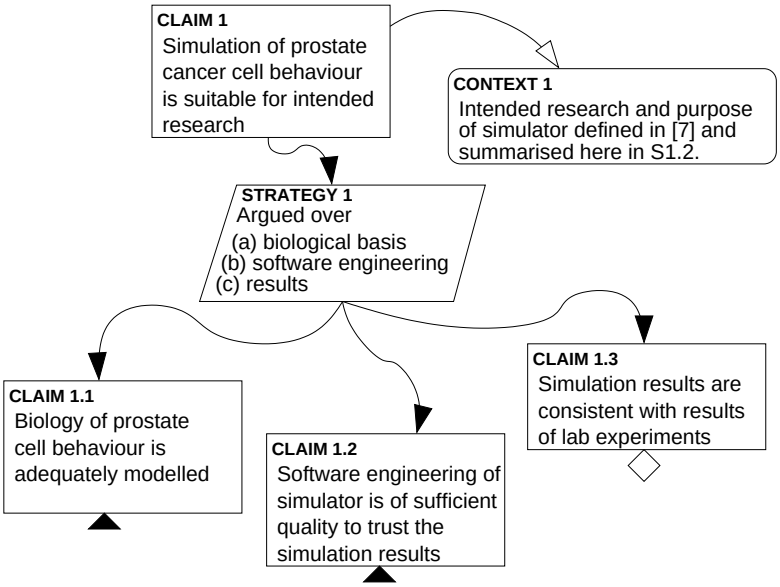

Fig. 2. A top-level argument for the adequacy of the simulation. Undeveloped claims that are expanded in other diagrams in this paper are indicated by filled triangles.

to Claim 1.3, consistency of simulation results and laboratory results, are discussed in [8], where a similar argument over results is presented, and [15], where Polack presents a generic argument over the results of a simulation.

2.1 Claim 1.1: adequate modelling of biology

The development of Claim 1.1, that the biology of prostate cell behaviour is adequately modelled, is shown in Figure 3. The context points to a definition of adequate modelling. Ultimately, the evidence of adequacy here is that the domain experts and developers have jointly reviewed all aspects of the domain model. The argument can systematically identify what needs to be reviewed and agreed: this leads here to two strategies. Note that, whereas in safety case argument, multiple strategies present complementary approaches, here Strategies 1.1.1 and 1.1.2 represent different parts of the argument.

Figure 3 shows claims arising from Strategy 1.1.1, arguing that identified cell types and transitions are modelled adequately. Context 1.1.1 again points to Droop et al [7] as the definitive description of the cell types and transitions in the domain model. We do not show the expanded claims here: Claim 1.1.1.1 (that the identified cell types provide a sufficient basis for the simulated system) and claim 1.1.1.2 (that the cell transitions are adequately modelled) are substantiated by systematically addressing each cell type/transition in turn and recording the evidence for its adequacy, for instance by reference to the minutes of meetings where each was reviewed by the developers and domain experts.

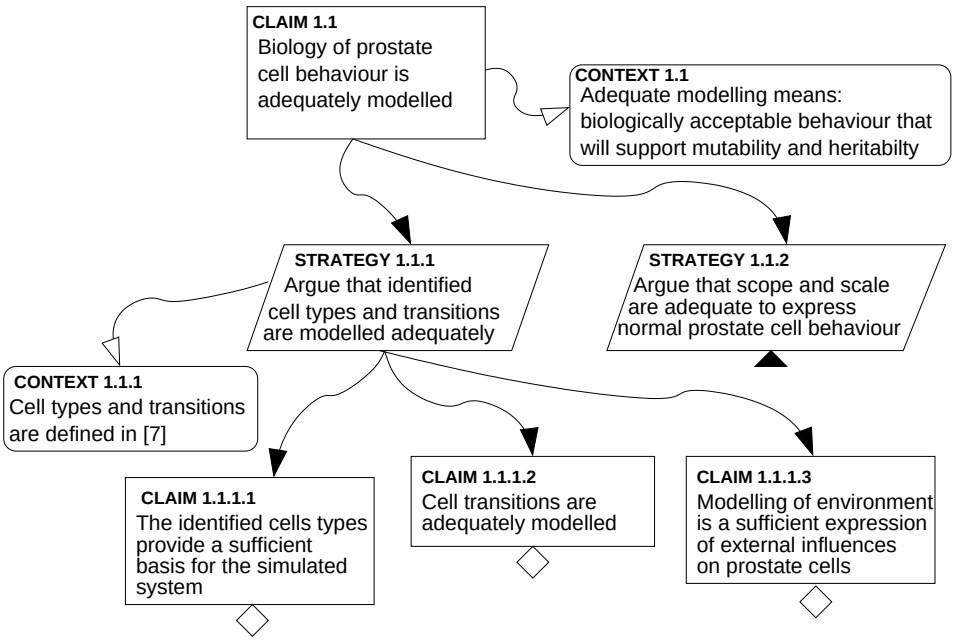

Fig. 3. Expanding Claim 1.1 (Figure 2) to consider adequate modelling of prostate cell behaviours (as described in [7]).

Claim 1.1.1.3, that the modelling of the environment is a sufficient expression of external influences on prostate cells, is more interesting: there is a potentially unlimited interaction between the environment and the prostate cells, encompassing direct interaction (with chemicals, temperature, radiation etc) and indirect representation of, for instance, spatial effects such as proximity and crowding. Here, the discussion between modellers and domain experts can be directed by focusing on potential constraints and triggers on transitions: critical systems engineering offers a range of deviational techniques for challenging models that could be used to reveal any hidden assumptions of transitions.

Figure 4 shows the development of Strategy 1.1.2, that the scope and scale are adequate to express normal prostate cell behaviour. Since the development of the prototype is not complete, Claim 1.1.2.1, that the numbers of cells in the simulation are sufficient for biologically realistic behaviour to emerge, cannot yet be elaborated: a strategy could propose suitable laboratory experiments and hypotheses for dual exploration using the completed simulator. Claim 1.1.2.2 demonstrates how a claim can be concluded with evidence: the evidence states that the claim, that modelling the prostate as a closed system of cells is biologically realistic, has been agreed.

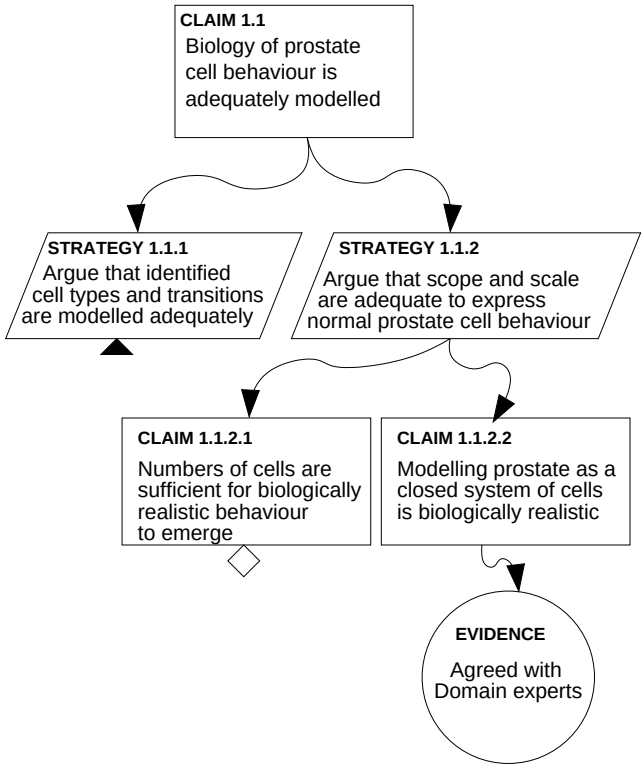

Fig. 4. Expanding Claim 1.1 (Figure 2) to consider adequate scope and scale of prostate cell models.

2.2 Claim 1.2: quality of software engineering

Claim 1.2 in Figure 2 concerns the quality of the software engineering of the simulation. The expansion (Figure 5) proposes the two-part strategy of arguing seamless development, and of demonstrating that the software is verified. Seamlessness is the process by which models are systematically refined or developed from abstract to code. Claim 1.2.1 makes reference to the CoSMoS process, since the principled approach to modelling and simulation recommends seamless development (see e.g. [17]); the claim could be addressed by a systematic review of how the domain model concepts map to concepts in the platform model. Similarly, Claim 1.2.2 requires us to demonstrate the mapping from the platform model to the JCSP code of the simulator. We cannot fully elaborate this claim until the coding – and verification – is complete.

Verification is covered by two claims. Claim 1.2.3 makes reference to the development environment for Java, and requires support for the statement that the (unit) testing and debugging facilities of the environment are sufficient to establish trust in the code: this is a pure software-engineering challenge for the

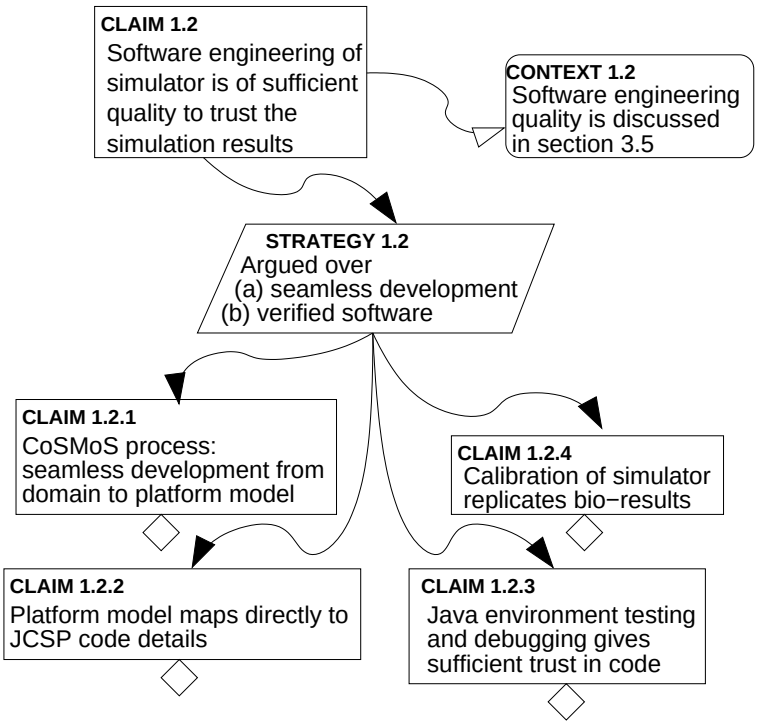

Fig. 5. Expanding Claim 1.2 (Figure 2) concerning Software Quality

developers. Claim 1.2.4 refers to calibration, the process in which the completed simulator is given biological data for a particular initialisation, and expected to replicate biological output data. Note that this claim is closely related to Claim 1.3, but is undertaken as part of the verification of the simulator, rather than as part of subsequent use of the simulator as a research tool.

3 Discussion

This section identifies issues in the use of argumentation for validation of fitness for purpose, illustrated in relation to the arguments for the cell division and differentiation validity in section 2.

3.1 Fitness for purpose

A validity argument presents a specific claim – usually that the simulator is suitable for the intended research. This is important: a simulation may not be a good model of its domain, but could be suitable for an intended research goal. The reason for using a simulator to study a complex system is to abstract away enough of the detail to allow insight: a simulation that is too faithful to its domain is almost as complex as that domain, and thus a poor aid to exploration and understanding.

A connotation of the need to validate fitness for purpose for intended research is that, if the research goal changes, the validity argument must be revisited. It is unlikely that the prostate cell simulator would be fit for the purpose of research on, say, a breast cell model. Even within the realm of prostate cell modelling, a laboratory with a different theoretical standpoint, or a different interpretation of the biology, might find that our simulator was not fit for exploring its hypotheses. The simulator expresses a scale and scope that matches the scope of the laboratory research of the domain experts. The tie-in of the simulator to the domain expert view means that, if the results of simulation are to be published to the wider prostate cancer community, it is particularly important to record the biological and theoretical basis of the simulation.

3.2 Living arguments

We have seen that a validity argument is specific to the domain experts' intended research and the purpose of the simulation. However, a validity argument also changes as the simulator development and simulation experimentation proceeds.

The argument outlined in Section 2 was developed before completion of the simulator: before the initialisation and running of any simulation. *A validity argument is a living argument*, and can help to direct the ways in which the simulation development proceeds. In our development, for example, the software engineering strategy (Claim and Strategy 1.2) focuses the attention of the developers on the quality requirements for the implementation (covered in more detail in Section 3.5): attention was paid to development of robust (manual) mappings from the domain model to the platform model and simulator implementation. Thus, the use we make of argumentation helps guide development of a simulator that is demonstrably fit for purpose, rather than simply recording an end-point opinion about fitness for purpose of the finished simulator.

Once a simulator is validated, it can be used for simulation experiments; to analyse the results of simulation, we use the research context, built up in the development and the process of validation. Through using the simulator

and conducting *in silico* experiments, we learn about the domain, and about the simulator, and can expand the validity argument with new evidence and understanding.

The validity argument is a snapshot relating to the simulation project on the day that it was produced. If we want to publish the simulation results, and expose the simulator to community evaluation, we need to ensure that the validity argument is up-to-date and matches exactly the published version of the results. In the prostate cell work, simulated experimentation results are intended to be backed up by laboratory results, so this level of rigour is not needed.

Since simulator validity is not absolute, it is important to know for whom the validity argument is created. The argument that is shown in Section 2 was created by the developers for the use of the developers and domain experts at this point in the development. The developers and domain experts are content that the simulator is suitable for the intended prostate research if the biological model is agreed to be adequate, the software engineering has sufficient quality, and the simulation results are consistent with those from the laboratory research. *You* may disagree with this position; the point is that the researchers (prostate cancer scientists and simulator developers) have made their position explicit. As the research progresses, the validity argument will develop to reflect changing understanding.

3.3 Reviewing validity

Whenever an argument is reviewed, it is likely to be expanded. In simulation validity, it is likely that review will also extend the research context and enable a better appreciation of the strengths and limitations of the simulator. This can be illustrated by describing an issue that arose when the authors reviewed the argument in Figure 4, during the writing of the paper; that is, after the domain experts and developers had agreed on the domain model, and had accepted that it was a valid model of the biology of prostate cell division and differentiation.

In Figure 4, Claim 1.1.2.2 states that *Modelling prostate as a closed system of cells is biologically realistic.* The evidence states the agreement of domain experts. In review, the term *closed system* was identified as contentious: complex biological systems are open by definition. It was first noted that we have not explained what we mean by a closed system of cells: this can be addressed by adding a context explanation: Context 1.1.2.2 would state that the agreed model comprises a precise set of cell types, their division and differentiation, and a limited (finite) set of environmental influences. Next, we need to identify what the domain experts agreed: the records of our meetings showed that it had been explicitly discussed and decided that the domain model did not need to model blood flow, nutrient supply and other biological fluxes in order to develop a simulation of the prostate cell model that is suitable for the intended research. We can then link the argument evidence to the precise meeting record. Either through such a link or through use of a GSN justification, we can also record the rationale for the domain experts' confidence that the closed cell model is sufficient.

The review of Claim 1.1.2.2 also leads to a clarification of the purpose and suitability of the simulator. Discussion of the implications of a "closed" system of cells identified a range of specific research questions that cannot be addressed with this simulator. Since the simulator does not include an explicit model of blood vessel formation and capillary bed structure, it is impossible to explore vascularisation and hypoxia effects in developing tumours, and research with the simulator cannot explore interventions that rely on blood flow or hypoxia [14, 6].

In the course of reviewing an argument, it is inevitable that people query the detail and the extent of an argument. In safe-systems engineering, such queries have to be taken very seriously and addressed before the safety case is accepted. However, in our more informal use of argumentation, it is sufficient that a consensus is reached on the adequacy of the argument; thus important observations on the adequacy of the argument are pursued and concluded. In work using the CoSMoS principled approach and informal validation of the sort described in this paper, we have not yet failed to reach consensus, but a notable feature is that the domain experts are often *more* trusting of the simulators than are the simulation developers. This observation needs following up in patterns of guidance for the construction and review of validity arguments in collaborative research.

3.4 Validity and the representation of the domain

To be fit for purpose, the domain model needs to be an abstraction that is agreed to be suitable for the intended research: the prior identification of explicit domain experts is essential to achieving this.

In an ideal simulator, each concept in the real domain would map directly to a concept in the simulator. However, as noted in section 3.1, a simulator that is too similar to its complex-systems domain is unlikely to be a good aid to understanding. In the prostate cell simulator, whilst it would be an interesting challenge to construct a simulator that modelled the biochemistry of signalling as well as cell division and differentiation, this would blur the focus on the scientific purpose and motivation of simulation. For the intended purpose of this simulator, it is not necessary to know how cells emit and receive chemical signals; it is sufficient to know that cells affect the environment, and that the environment affects cells, through particular interactions and behaviours. The model of the environment must include the appropriate parameters to implement cell-environment interaction, guided and checked by the domain experts. If the argument that the simulator is sufficient were extended, the validation of each of these areas would be added; we might also add a separate set of claims relating to the cross-validation issues.

In addition to identifying the level of abstraction that is appropriate to a domain and a simulation purpose, the domain experts work with the developers to determine how to model necessary parts of the domain that are not well understood, or are not easily amenable to measurement.

The prostate cell model abstracts away from low-level detail (biochemistry, physics, the complex internal structure of cells, etc.), ignores concepts other than the identified cell types and transitions, and simplifies other concepts – cells are represented as a "genome", a pre-defined set of data defining the cell's (computational) state. The modelled cell types map well to biologically distinct (stem and luminal) or distinguishable (committed basal and transit amplifying) cell types. The domain model also introduces the entirely-artificial concept of a daughter cell, to allow separation of cell division and cell differentiation processes in the models. The daughter-cell concept was discussed at length with biologists, and agreed as a reasonable way to represent biological behaviour. The level of abstraction avoids many of the biological uncertainties in the detail of cell mechanisms. This unusual level of clarity makes the validation of the biological modelling straightforward (in comparison to many research domains).

In mapping from the domain model to the platform model, the developers need to identify implementation structures and remove any emergent or derived behaviours included in the domain model (these need to be consequences of computation, and must not be built into the simulator). The demands of computation necessarily add features that have no obvious dual in the domain. Some computational features are common in simulations of complex systems, and could be the subject of generic analysis, and of validation patterns. Three examples are as follows: (a) computer simulation using digital representations of continuous aspects (time, space, gradients, differentiation, etc.); (b) parallel implementation using artificial synchronisation on, for example, time or the completion of a task (e.g. using barriers over channels); (c) computer simulation initialised to some artificial starting point, where the domain is a continuous ongoing behaviour.

The prostate cell simulator is a parallel implementation, in which both time and differentiation/division are digitised. The effects of digitising time need wider study, but the representation of the continuous differentiation/division of cells as transitions between specific cell states has been discussed and agreed to be adequate. In relation to initialisation, the prostate cell model can be started from an "embryonic" state (a small number of stem cells) and the simulation can be used to populate a prostate with cells; indeed, this is the focus of the first phase of the project, with the ability to populate a naive prostate calibrated against laboratory data.

As a last point, the representation of the domain introduces the surrogacy problem: each concept in the simulator will have explicit mappings to domain concepts, but also plays some part in representing the things that are not explicitly represented. Surrogacy is not simply an abstraction issue (e.g. the cells in the simulation surrogate the biochemistry), and it applies to biological measurement as well as to simulator concepts. Read et al [19] consider the surrogation problem in calibration and sensitivity analysis, as well as results interpretation. In the prostate cell simulator development, for example, the modelled cell behaviours surrogate behaviours that are due to other prostate components, whilst the biological data on division rates implicitly includes the effect of crowding. In relation to cell crowding, the easiest model to validate would be one with a

direct mapping between biological space and simulator space; this would produce simulation results on spatial distribution and dynamics. Unfortunately, the realistic 3D space model is not so easy to relate to the biologically-observable data, since there are only very limited ways in which researchers can observe the internal dynamics of a real prostate (this is one motivation for turning to agent-based simulation as a research tool). The representation of crowding in the simulator is thus through adjustment to transition parameters and/or environmental inputs, and needs careful validation through calibration and, possibly, through comparison of spatial and aspatial versions of the simulation.

In practice, it is impractical to try to enumerate all the issues that affect the mapping of domain concepts through to implementation. Furthermore, for many mappings, the consequences of the design decisions are hard to analyse or assess. Thus, the practical validation requirement is to record the identified design decisions, assumptions, omissions, abstractions and simplifications, and any known effects on the simulation. This allows an informed assessment of results. For a critical simulation (if someone were ill-advisedly to attempt to use the simulation evidence unsupported by laboratory evidence, to develop a new cancer intervention, perhaps), we would need to take more care in identifying, and analysing the effects of these issues. To date, we have not undertaken such a critical or high-impact simulation, but we have identified a range of critical systems engineering techniques that can help to challenge models and identify assumptions [17].

3.5 Validity and simulation engineering

The engineering quality of complex systems simulation does not receive much attention in scientific literature, though there is some coverage in conventional simulation literature (e.g. [4]). In complex domains, initiatives on documenting scientific simulation, such as ODD [5, 10], focus on making code and parameter settings available, not on ascertaining whether the model is rational and the code verified. We cannot easily measure the quality of software, especially where the software implements a complex system; we need alternative ways to develop confidence in the quality of the simulator as an artifact. One important issue concerning software quality is the need to understand whether observed (e.g. emergent) behaviours of the simulation are artifacts of the implementation or consequences of the represented domain concepts.

The validity argument strategy (Strategy 1, Figure 2) makes the need to validate the implementation explicit, whilst the elaboration of Claim 1.2 (Figure 5) makes the approach taken to software quality explicit. An advantage of the argumentation approach is that explicit statements are open, and can thus be challenged or discussed.

The rationale for the strategies used here in respect to software engineering validation is based on an understanding of software engineering methods. Conventional methods work from known requirements that have been discussed and agreed with clients. Here, requirements relating to the subject of simulation are addressed by domain modelling, and validated in the expansion of Claim

1.1. The software engineering strategy relates to quality requirements: that the implementation is fit for purpose and the results can be understood in the context of the domain. This requires us to validate the implementation against the domain model. Conventional software engineering proposes *seamlessness* (described briefly in Section 2.2) as a traceable way of moving from abstract models to code. Seamlessness is aided by adhering to the same paradigm and semantics for modelling and implementation, thus reducing opportunities for misinterpretation. The CoSMoS process (described in Section 1.2) proposes seamless development from domain model to platform model to simulator implementation. Thus, for instance, Claim 1.2.1 appeals directly to our use of the CoSMoS process to support the claim of seamless development from domain to platform models. A similar style, of appeal to a particular development method, is used when considering the artifact quality in safety case argumentation. More specifically, in [7], we describe a reasonably seamless implementation process from the domain models, in which we amend the semantics of diagrammatic modelling notations to provide a good representation of the domain *and* a clean mapping to the code design. A complete argument of software engineering quality would systematically address each concept in the abstract (domain) models and show how these are seamlessly developed. The argument would be strengthened if the seamless approach were supported by model-driven engineering techniques, and specifically by developing and supporting domain specific languages.

Note that CoSMoS (see section 1.2) uses the process of domain modelling to cement the relationship between developers and researchers (the clients), to clarify the scope and levels of the simulation, and to determine the purpose of simulation. Close collaboration is an important contributor to quality, as it opens the development models and process to continual challenge. Challenges come from both developers and domain experts; developers should highlight inadequacies of the software, as well as asking naive questions about the domain.

The second aspect of software engineering in Strategy 1.2 is verification: the demonstration that the system works as intended, through testing and/or formal analysis. Testing of complex systems simulations is interesting, since, even if we know what results are expected (which is not always the case), the results are often non-deterministic: multiple runs and statistical analysis of data results are needed to establish whether, with some likelihood, the results of the simulation are in line with those from laboratory experiments. In the validity argument summarised here, there is thus a close link between the software engineering claims and Claim 1.3 concerning the consistency of results from the domain and the simulator.

For conventional software testing, programming environments (e.g. Eclipse or NetBeans for Java) provide programming support such as built-in debugging and analysis tools: in the prostate cancer model, use of Java and the JCSP library means that verification support is available. However, more efficient process-oriented languages such as occam-π lack environmental support. In partial compensation, occam-π has compiler support for safe programming idioms (avoiding deadlock). The foundations of JCSP and occam-π in the CSP formal language

make it amenable to some formal analysis of programs – protocols are particularly well suited to CSP analysis.

Neither formal analysis nor testing is sufficient to eliminate errors in design and implementation; they must be complemented by calibration [19]. There can be a fine line between a desired emergent behaviour and the consequences of a bug in a design or a program; the simulation developers as well as the domain experts must be confident that the observed behaviours are "valid", not artifacts of a faulty simulator.

A common problem with calibration is achieving appropriate simulation scales. We would investigate whether we need biological-scale numbers, or whether it is sufficient to simply to initialise a simulation with biologically-realistic proportions of each type of cell – in other words, can we achieve biologically-valid results on systems with the right proportions of components but the wrong numbers of components? Emergence of desired behaviour is sometimes scale-dependent, so this is an important area of analysis. In the prostate cell modelling, an additional factor relating to scale and calibration is the eventual focus on cancer-causing mutability and heritability: the agent-based simulation approach to simulation was chosen specifically to allow modelling of rare mutation, and a valid simulator must have sufficient numbers of cells to accommodate low probability events.

The activity of calibration draws on the whole development and documentation of the simulator, and fundamentally relies on the close collaboration built up in the development of the simulator. It must also take account of the validity of calibration data. Laboratory data may be from many specimens, different tissue types, even different species, measured to differing levels of accuracy. In cell-level systems, it is usually possible to estimate numbers and proportions of cells, and, to some extent, how these values change over time: for the prostate cell model, we do have biologically-robust data. However, the domain experts and developers need to be constantly alert to data problems (e.g. data from human patients vs. data from laboratory cell-lines).

It is worth noting that the calibration of our simulator will, in itself, achieve the purpose of the first phase of the simulation project: to replicate observed proportions of cells in the normal prostate (Section 1.2). Once calibration is complete, we intend to construct a calibration argument pattern.

4 Summary and Conclusions

We have presented aspects of the validation of the prostate cell model simulation (described in [7]), showing how an understanding of the fitness for purpose and limitations of a simulation used as a research tool can be captured. The meaning of validation in the context of complex system simulation is summarised, along with the argumentation approach used here.

The argument structure is summarised using GSN, and represents the mutual understanding of the sufficiency of the simulation at a particular point in the development; the argument will be developed as the study proceeds. This

argument is specific to this simulator development for this domain, this group of domain experts and these developers.

4.1 Further work

The immediate next steps in this work are to complete the recording of the fitness for purpose argument, as described in Sections 2 and 3.2. The argument will develop as the simulator development and calibration progresses, and as the research moves into exploration of the prostate cell division and differentiation and cancer neogenesis.

Our research on validity argumentation uses the prostate cell simulation and other complex systems simulations (e.g. [9, 8])) to establish the commonalities and differences between validity arguments and established argumentation uses such as safety case and dependency argumentation. In particular, we can draw on experience of safety case consistency, and recent work on the justification of strategies, as well as work on dependency across claims. We also intend to develop tool support for simulation validation, focusing on argumentation structures and on linkage to supporting evidence, context, justifications and assumptions. Tool development will provide a definition of the variant GSN notation used for validity arguments, along with guidance on constructing and reviewing validity arguments.

As the body of simulation validation work grows, it is apparent that there are many generic issues to be addressed: for example, Section 3.4 has identified the problems of discretisation of continuous features such as time, space, gradients, and related problems such as the difficulty of matching spatial simulation results to aspatial data (see also [2]). As generic problems are identified, they need analysing, to understand what effects they have on the accuracy of simulation, and whether the effects are general or specific to certain sorts of simulation or questions explored through simulation. We anticipate that argument patterns will help document the connotations for validity of these generic problems.

Acknowledgements

This work is supported by a Program Grant (to N. J. Maitland) from Yorkshire Cancer Research, by TRANSIT (EPSRC grant EP/F032749/1) through the York Centre for Complex Systems Analysis, and by CoSMoS (EPSRC grant EP/E053505/1). We would like to thank the members of the Cancer Research Unit in York for their invaluable input of time and expertise.

References

1. R. Alexander. *Using Simulation for Systems of Systems Hazard Analysis.* PhD thesis, Department of Computer Science, University of York, 2007.
2. P. S. Andrews, F. Polack, A. T. Sampson, J. Timmis, L. Scott, and M. Coles. Simulating biology: towards understanding what the simulation shows. In *Workshop on Complex Systems Modelling and Simulation*, pages 93–123. Luniver Press, 2008.

3. P. S. Andrews, F. A. C. Polack, A. T. Sampson, S. Stepney, and J. Timmis. The CoSMoS Process, version 0.1. Technical Report YCS-2010-450, Dept of Computer Science, Univ. of York, 2010. www.cs.york.ac.uk/ftpdir/reports/2010/YCS/453/YCS-2010-453.pdf.
4. Osman Balci. Verification, validation, and accreditation. In *WSC '98*, pages 41–48. IEEE Computer Society Press, 1998.
5. U. Berger, C. Piou, K. Schiffers, and V. Grimm. Competition among plants: Concepts, individual-based modelling approaches, and a proposal for a future research strategy. *Perspectives in Plant Ecology, Evolution and Systematics*, 9:121–135, 2008.
6. D. Bishop-Bailey. Tumour vascularisation: a druggable target. *Current Opinion in Pharmacology*, 9:96–101, 2009.
7. A. Droop, P. Garnett, F. A. C. Polack, and S. Stepney. Multiple model simulation: modelling cell division and differentiation in the prostate. Accepted for CoSMoS Workshop, 2011.
8. T. Ghetiu, R. D. Alexander, P. S. Andrews, F. A. C. Polack, and J. Bown. Equivalence arguments for complex systems simulations - a case-study. In *Workshop on Complex Systems Modelling and Simulation*, pages 101–140. Luniver Press, 2009.
9. T. Ghetiu, F. A.C. Polack, and J. Bown. Argument-driven validation of computer simulations – a necessity rather than an option. In *VALID*, pages 1–4. IEEE, 2010.
10. V. Grimm. Ten years of individual-based modelling in ecology: what have we learned and what could we learn in the future? *Ecological Modelling*, 115(2-3):129–148, 1999.
11. R. Hawkins, T. Kelly, J. Knight, and P. Graydon. Safety cases – a new approach to creating clear safety arguments. In *SSS'11*, pages 3–23. Springer, 2011.
12. P. Humphreys. *Extending Ourselves: Computational Science, Empiricism, and Scientific Method*. Oxford University Press, New York, 2004.
13. T. P. Kelly. *Arguing safety – a systematic approach to managing safety cases*. PhD thesis, Department of Computer Science, University of York, 1999. YCST 99/05.
14. S. Kizaka-Kondoh, M. Inoue, H. Harada, and M. Hiraoka. Tumor hypoxia: A target for selective cancer therapy. *Cancer Science*, 94(12):1021–1028, 2005.
15. F. A. C. Polack. Arguing validation of simulations in science. In *Workshop on Complex Systems Modelling and Simulation*, pages 51–74. Luniver Press, 2010.
16. F. A. C. Polack, P. S. Andrews, T. Ghetiu, M. Read, S. Stepney, J. Timmis, and A. T. Sampson. Reflections on the simulation of complex systems for science. In *ICECCS*, pages 276–285. IEEE Press, 2010.
17. F. A. C. Polack, P. S. Andrews, and A. T. Sampson. The engineering of concurrent simulations of complex systems. In *CEC*, pages 217–224. IEEE Press, 2009.
18. F. A. C. Polack, T. Hoverd, A. T. Sampson, S. Stepney, and J. Timmis. Complex systems models: Engineering simulations. In *ALife XI*, pages 482–489. MIT press, 2008.
19. M Read, P. S. Andrews, J. Timmis, and V. Kumar. Techniques for grounding agent-based simulations in the real domain: a case study in Experimental Autoimmune Encephalomyelitis. *Mathematical and Computer Modelling of Dynamical Systems*, 2011. accepted.
20. R. G. Sargent. Verification and validation of simulation models. In *37th Winter Simulation Conference*, pages 130–143. ACM, 2005.
21. R. A. Weaver. *The Safety of Software – Constructing and Assuring Arguments*. PhD thesis, Department of Computer Science, University of York, 2003. YCST-2004-01.

22. M. Wheeler, S. Bullock, E. Di Paolo, J. Noble, M. Bedau, P. Husbands, S. Kirby, and A. Seth. The view from elsewhere: Perspectives on ALife modelling. *Artificial Life*, 8(1):87–100, 2002.
23. S. P. Wilson, J. A. McDermid, C. H. Pygott, and D. J. Tombs. Assessing complex computer based systems using the goal structuring notation. In *ICECCS*, pages 498–505. IEEE Computer Society, 1996.