Proceedings of the 2011 Workshop on
Complex Systems Modelling and Simulation

CoSMoS 2011

Susan Stepney, Peter H. Welch,
Paul S. Andrews, Carl G. Ritson,
Editors

CoSMoS 2011

Luniver Press
2011

Published by Luniver Press
Frome BA11 6TT United Kingdom

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library

CoSMoS 2011

ISBN-10: 1-905986-32-7
ISBN-13: 978-1-905986-32-3

Preface

Building on the success of previous CoSMoS workshops, we are pleased to be running the forth CoSMoS as a satellite event of the 2011 European Conference on Artificial Life (ECAL11) in Paris, France. ECAL11 is an especially good fit for the CoSMoS workshop, examining critical properties of living and life-like systems and attracting a broad range of interdisciplinary researchers. The systems examined by these researchers are inherently complex, and various modelling and simulation techniques have become key to exploring and understanding their properties.

The genesis of the CoSMoS workshop is the similarly-named CoSMoS research project[1], a four year EPSRC funded research project at the Universities of York and Kent in the UK. The project aims are stated as:

> The project will build capacity in generic modelling tools and simulation techniques for complex systems, to support the modelling, analysis and prediction of complex systems, and to help design and validate complex systems. Drawing on our state-of-the-art expertise in many aspects of computer systems engineering, we will develop CoSMoS, a modelling and simulation process and infrastructure specifically designed to allow complex systems to be explored, analysed, and designed within a uniform framework.

As part of the project, we are running annual workshops, to disseminate best practice in complex systems modelling and simulation. To allow authors the space to describe their systems in depth we put no stringent page limit on the submissions.

The focus for the fourth CoSMoS workshop is on how complex systems simulations can begin to approach the scale of real-world complex systems. To complement this theme, our invited keynote speaker was Prof. Dirk Helbing from ETH Zürich, Switzerland, who is the scientific coordinator of FuturICT flagship, which aims "to understand and manage complex, global, socially interactive systems, with a focus on sustainability and resilience"[2].

The main session of the workshop is based on six full paper submissions:

Andrews et al. summarise the modelling concepts of the CoSMoS approach to complex systems simulations, and describe how these mod-

[1] The CoSMoS project, EPSRC grants EP/E053505/1 and EP/E049419/1, http://www.cosmos-reseach.org

[2] http://www.futurict.ethz.ch/FuturICT

els can be related via their common metamodel, and how CoSMoS can be applied to engineering bio-inspired systems.

Evora et al. present a framework for simulating complex energy systems, and describe an agent-based model and simulation of household electricity demand which is used to investigate the load curve of 1000 households composed of different social groups.

McEwan et al. show how complex mathematical models can be expressed using techniques for specifying finite state machines, and how this enables a systems approach to be used at a coarser granularity than typically seen in systems biology.

Guest et al. explore the benefits and pitfalls of visualisation as a tool to aid understanding of complex systems, presenting a case-study of agent communication on the intra-individual, inter-individual and community scales.

Droop et al. describe a hybrid multiscale modelling approach based on Petri net and object-oriented models, which is applied to a case-study on prostate cell division and differentiation.

Polack et al. complement the paper by Droop et al. by investigating the validity of the prostate cell simulation, and discussing more general issues of validating complex systems simulation for scientific research.

We also invited authors to submit abstracts, for presentation in a poster session. Abstracts for the following posters are presented in the proceedings:

Mancy et al. develop an alternative multiscale pairwise approximation approach for ecological modelling, and compare to a spatially explicit multiscale simulation.

Alden et al. present an agent-based simulation of Peyer's Patch formation developed following the CoSMoS approach, and explore how the simulation results relate back to observation of the real system.

Eleftherakis et al. propose a framework to enable the engineering of emergent properties into artificial distributed networks, combining classical software engineering practices with multi-agent systems.

Clayton et al. explore the Go programming language for developing cellular automata type simulations, comparing Go to more established CSP based concurrent languages.

Our thanks go to our keynote speaker and to all the contributors for their hard work in getting these papers, abstracts and posters prepared and revised. All submissions received multiple reviews, and we thank the programme committee for their prompt, extensive and in-depth reviews. We would also like to extend a special thanks to the organising

committee of ECAL11 for enabling our workshop to be co-located with this conference. We hope that readers will enjoy this set of papers, and come away with insight on the state of the art, and some understanding of current progress in complex systems modelling and simulation.

Programme Committee

Rob Alexander, University of York, UK

Paul Andrews, University of York, UK

Fred Barnes, University of Kent, UK

James Dyke, Max Planck Institute for Biogeochemistry, Germany

George Eleftherakis, CITY College, International Faculty of the University of Sheffield, Greece

Teodor Ghetiu, University of York, UK

Tim Hoverd, University of York, UK

Colin Johnson, University of Kent, UK

Enrique Kremers, European Institute for Energy Research, Karlsruhe, Germany

Nick Owens, National Institute for Medical Research, UK

Rebecca Mancy, University of Glasgow, UK

Fiona Polack, University of York, UK

Carl Ritson, University of Kent, UK

Adam Sampson, University of Abertay, Dundee, UK

Susan Stepney, University of York, UK

Jon Timmis, University of York, UK

Peter Welch, University of Kent, UK

Table of Contents

CoSMoS 2011

CoSMoS process, models, and metamodels

Paul S. Andrews[1], Susan Stepney[1], Tim Hoverd[1], Fiona A. C. Polack[1], Adam T. Sampson[2], and Jon Timmis[1]

[1] YCCSA, University of York, UK, YO10 5DD
[2] Institute of Arts, Media and Computer Games, University of Abertay Dundee, UK, DD1 1HG

Abstract. We summarise the existing CoSMoS approach to modelling and simulating complex systems, then introduce how the various CoSMoS models are related via their metamodel, and demonstrate the generality of the process by discussing its application to engineering bio-inspired systems.

1 Introduction

The CoSMoS project is developing tools and techniques that enable the construction and exploration of simulations for the purpose of scientific research. This is a necessarily interdisciplinary endeavour between scientists who study a particular domain (the *domain experts*), and software engineers who construct simulations to facilitate the study of that domain (the *developers*). Together, the domain experts and developers are involved in open-ended scientific research: the simulations are used as a tool to support theory exploration, hypothesis generation, and design of real-world experimentation [10, 11].

Our work is driven by a series of simulation case-studies from a broad range of disciplines (immunology, ecology, and sociology), with a focus on simulating *complex systems*. Such systems are amenable to computer simulations, displaying high-level system behaviours that emerge as a consequence of many simple behaviours at a lower level, where the high-level behaviour cannot be readily deduced as a simple combination of the low-level behaviours. A classic example of a complex system behaviour is bird flocking: the individual behaviour of birds flying together can result in the emergence of a flock at the population level. Computer simulation allows us to examine such systems from a *holistic* point of view.

To run computer simulations we need to engineer a simulation platform. This requires us to explicitly represent some knowledge of the

system being studied in a form that can be implemented on a computer. This representation, the *source code*, is either designed manually by the developers or automatically generated from a higher-level description. In many existing approaches the source code is the only explicit description of the aspects of the target domain that are being simulated. Source code contains numerous implicit assumptions[3] concerning both the scientific aspects of the work, and the engineering design of the simulation platform.

To mitigate inappropriate assumptions in the design of simulation platforms, and to have greater confidence that simulation results can actually tell us something that relates to the real system being studied, we use a series of related models to drive and describe the development of the simulation platform and simulation results generated from its use. Systematic development assists interaction between domain experts and developers, and improves our confidence in, and interpretation of, the results of simulations [12].

Here we summarise the existing CoSMoS modelling process[4], and introduce the CoSMoS metamodel. In §2 we describe the existing CoSMoS modelling approach in terms of its various models. In §3 we introduce a metamodel and explain how it relates these CoSMoS models. In §4 we demonstrate the generality of the CoSMoS process by discussing its application to engineering bio-inspired systems.

2 The CoSMoS process

Our approach uses the following concepts [1]: *domain*, *domain model*, *platform model*, *simulation platform*, and *results model*. The domain represents the real-world system (or part of a system) that is being investigated. This is related to the models and simulation platform as shown in figure 1, where the arrows denote a flow of information from one concept to the next. Like all scientific work, the five development concepts are shaped by the *research context*.

Research Context: identifies the overall scientific context and scope of the simulation-based research being conducted. The scientific context can be elucidated by recording high-level motivations or goals, research

[3] An assumption is any kind of abstraction, simplification, axiom, idealisation or approximation.

[4] Project documentation of simulation, modelling and process descriptions [1, 11, 12], of validation and argumentation [5, 6, 10], of various biological system simulation case studies [2, 3, 4, 13], and of the CoSMoS workshop proceedings [15, 17, 18], is available from the CoSMoS project website www.cosmos-research.org

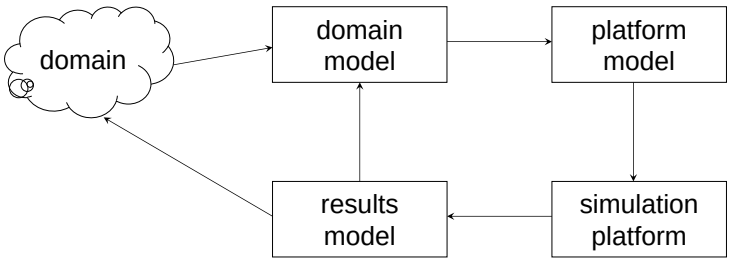


Fig. 1. Relationship between models, domain and simulation platform, where arrows represent flows of information. These are all framed by the research context.

questions, hypotheses, general definitions, and success criteria (how will you know the simulation has been successful). It is important to identify when, why and how these change throughout the course of developing and using the simulation. The scope of the research determines how the simulation results can be interpreted and applied. Importantly, it captures any requirements for validation and evaluation of simulation outputs. Consideration should be made of the intended criticality and impact of the simulation-based research, and if these are judged to be high, then an exploration of how the work can be validated and evaluated should be carried out. In summary, the role of the research context is to collate and track any contextual underpinnings of the simulation-based research, including the scientific background, and the technical and human limitations (resources) of the work.

Domain Model: explicitly captures understanding of the domain, identifying and describing the structures, behaviours and interactions present in the domain at a level of detail and abstraction suitable for addressing any identified research questions to be posed of the simulation platform. It is a model based on the science as presented by the domain experts, and its design should be free from simulation platform implementation bias; it separates the model of the science from the implementation details of the simulation platform. Importantly, the domain model is developed with the domain experts, and is used as a tool to exchange and discuss domain understanding between developers and domain experts. The domain model forms the agreed scientific basis for the eventual simulation platform.

Platform Model: an engineering derivation from the domain model, and a step towards construction of the simulation platform. The model is shaped by engineering design decisions, detailing the implementation of the structures, behaviours and interactions identified in the domain

model in a way that naturally translates to simulation platform technologies. This might dictate that some concepts in the domain model are abstracted or simplified, to allow efficient implementation. Some high-level emergent behaviours identified in the domain model are removed from the platform model, if the purpose of the research is to investigate the emergence of these behaviours from other model components. In general, given a hypothesis under consideration, components in the domain model that are *outcomes* of hypothesised mechanisms should not appear in the platform model: the answer should not be explicitly coded into the simulation platform, but must appear in some model. The platform model also adds instrumentation and interfaces to allow observation (visualisation), user interaction, and recording of the eventual results of using the simulation platform.

Simulation Platform: encodes the platform model in software and hardware platforms with which simulations can be performed. The simulation platform defines a set of parameters (variables) that allow the encoded model to be manipulated. The parameters are derived from the domain model and interpreted through the platform model, thus making the simulation platform accessible to domain experts with knowledge of the domain model.

Results Model captures understanding of the simulation platform based on the output of simulation runs, and provides the basis for interpretation of what the simulation results show. Its relationship to the simulation platform is analogous to the relationship between the domain and domain model. The results model is constructed by experimentation and observation of simulations, and might record observations, screenshots, dynamic sequences, raw output data, result statistics, as well as qualitative or subjective observations. The contents of the results model are compared to the domain model to establish whether the simulation platform provides a suitable representation of the real-world domain being investigated. The results model might also provide details to develop new experimentation, either on the simulation platform or in the real domain.

The domain, platform, and results models are generated and updated throughout simulation-based research: establishing the scientific basis, developing the simulation platform, and using the simulation platform to explore the domain. The models are used as devices to capture, communicate and reason about different aspects of the construction and use of the simulation platform, and how this relates to the domain. This includes annotating the appropriate model with scientific or engineering assumptions. By using a principled approach to simulation, the research

is ultimately open to review and challenge, and provides a basis for scientific reproducibility.

3 Models and Metamodels

The CoSMoS process emphasises the need for separate domain, platform, and results models. In this section we describe how these three models are related, and how they differ, in terms of metamodelling. The discussion here assumes that the modelling approach is agent-based, and that the research context is investigating emergent properties. Other modelling approaches and other research contexts are also possible: the specifics change, but the overall approach remains the same.

A model such as used in CoSMoS provides the abstract language of the relevant concepts; the platform model captures the concepts to be implemented in the (simulation platform) code. A metamodel provides the analogous language for writing a *model*: it defines the kinds of things that can occur in the model (it is the model of the model) [8, ch.8].

A metamodel can provide a rigorous link between different yet related models. For example, in agent-based modelling of systems with emergent properties, the metamodel includes concepts such as Agent, Rule, and Emergent. For a different style of model, such as an ODE model, the metamodel includes different concepts, such as Concentration and RateOfChange. An agent-based model of ant pheromone trails would include classes such as: Ant, an instance of Agent; and Trail, an instance of Emergent. An agent-based model of bird flocking would include classes such as: Bird, an instance of Agent; and Flock, an instance of Emergent.

The metamodel for the CoSMoS domain and results models must be essentially the same: the results model is constructed in the same language as the domain model, to enable direct comparison of the two, and so that the results are presented in a domain-relevant language. The platform metamodel differs in that it does not include the hypothesised emergent properties, but it does include interface and instrumentation concepts.

The components of an agent-based domain and results metamodel could include:

- Agent: the types of the entities in the agent-based model (eg, birds)
- Environment: the environment within which they act (eg, obstacles, gravity, wind)
- Rule: the agents' behavioural rules, including how they interact with each other (eg, flocking rules), and with their environment (eg, obstacle avoidance rules)

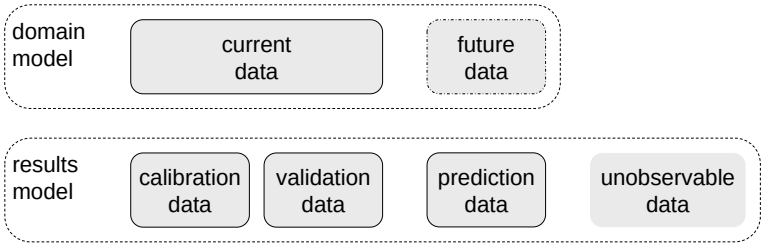


Fig. 2. The relationship between data in the domain and results models.

- Emergent: emergent properties exhibited by the dynamics of the model (eg, flocking)
- Data: scientific data used to quantify the model (eg, bird positions and velocities)
- Measure: a quantitative measure calculated from the data (eg, an entropy-based flocking statistic)

The domain and results models share a common metamodel. This does not mean that the two models are identical; it means that they are cast in the same language. The domain model has instances of metamodel concepts that capture specific domain concepts; the results model has instances of simulation analogues of those domain concepts. So where the domain model has Bird, the results model has Boid, the simulation analogue of Bird; both are instances of the metamodel concept Agent. The results model has Data instances, which stand in the same relation to its Agent instances as they do in the domain model (so if the domain model has bird positions and velocities, the results model has the corresponding boid positions and velocities). This means that the Measure instances can be essentially identical, allowing a direct comparison of the models *in domain terms*, see figure 2. The results model needs data so that it can be calibrated suitably; it needs further data so that it can be validated against the domain model (this is analogous to training and test data in machine learning [7]). At this stage it can be used to analyse data from novel scenarios, to make predictions; the domain can be augmented with new experimental data to test those predictions. Data in the results model that is not observable (even indirectly, through surrogates, or by investigating predictions) in the domain model is of little use. The necessity for suitable data in the results model implies requirements on the platform model.

The platform metamodel is different from that of the domain and results metamodel. In particular, it has no Emergent concept; it is im-

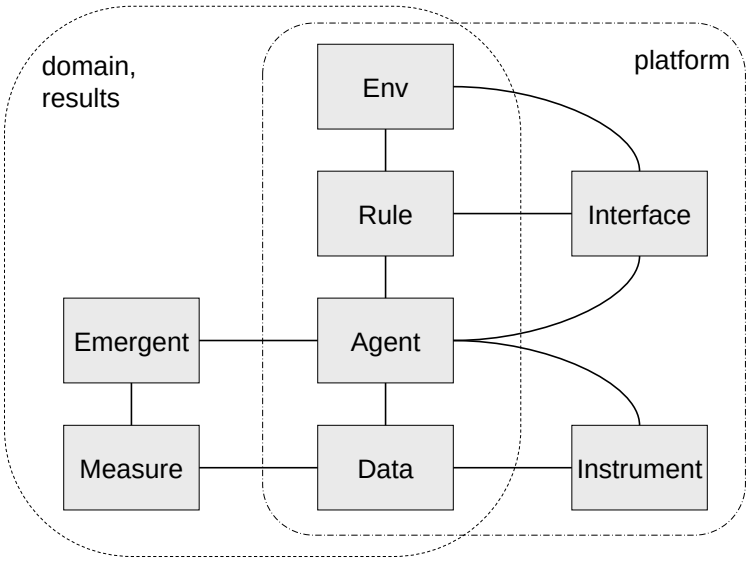


Fig. 3. A metamodel for agent-based simulation of emergent properties. The dashed boxes indicate the components of the metamodel that are used to describe the domain and results models, and the platform model.

portant that there is no way to program the desired answer into the simulation: it must emerge. (If the research context is not concerned with emergent properties, but some other kind of property, it is equally important to ensure that this does not get programmed in to the simulation.) The platform model adds Interface concepts, to allow user interaction with the simulation, and Instrumentation concepts, to allow extraction of the Data. The Data instances are common to the platform model and results model: that are extracted from the simulation, and analysed in the results model.

Although it is important that the domain and platform metamodels are different, it is also important that they share many concepts, allowing a common language. We define a single metamodel sufficient for all the individual models, and indicate which parts of it are specific to particular models, and which parts are common across the models. A schematic view of this metamodel is shown in figure 3.

A schematic view of part of a possible model is shown in figure 4. Not all the concepts in the metamodel need be instantiated as *classes* in the model. For example, Rule might be instantiated as rules of interaction,

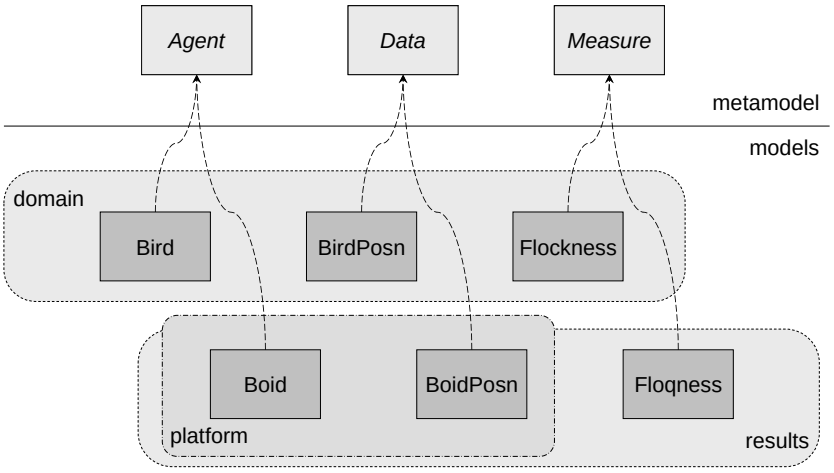


Fig. 4. The domain, platform, and results models for agent-based simulation of emergent flocking properties. The dashed arrows represent instance relationships between model concepts and their metamodel concepts. The dashed boxes show the components of the models: the domain has different concepts from those shared by the platform and results models.

as behaviours of boids; Env might be instantiated as a system, of several classes and methods.

4 CoSMoS process applications

The CoSMoS process described in §2 is a minimal process, suitable for building scientific simulations of real world domains. The underlying concepts of different models, and of a common metamodel, are generic, however, and can be used in a wider range of applications. Here we illustrate this by discussing their use for designing a class of bio-inspired algorithms (where the real-world domain is not being investigated for its own reasons, but is being used for inspiration), and for engineering

a bio-inspired domain (where the engineered domain is not pre-existing, but is being built as part of the process).

4.1 The conceptual framework for bio-inspired algorithms

Some of us have previously described a conceptual framework [16] for the principled development of bio-inspired algorithms. This conceptual framework (figure 5) describes a process of modelling the biology, abstracting out principles, and instantiating those principles as computational concepts (rather than building a direct, and naive, analogue of the biology). Probes (observations and experiments) are used to provide a partial view of the complex biological system. From this we build and validate simplified abstract representations of the biology. From these biological models we build analytical computational frameworks. These frameworks provide principles for designing and analysing bio-inspired algorithms applicable to non-biological problems, possibly tailored to a range of problem domains, and contain as much or as little biological realism as appropriate.

The conceptual framework is consistent with the CoSMoS modelling approach, with the identification of the components as shown in figure 6. The domain is the biological system from which we wish to draw our inspiration. The domain model is a suitable model of this biological system, that will form the basis for the bio-inspired algorithm design. We then abstract this model into a metamodel, capturing the relevant concepts and relationships of the biological model. At this meta-level, we can abstract away contingent details of the biology that are of no relevance to an algorithm. We then develop a separate instantiation of this same metamodel in the computational domain: the platform model. Thus the bio-inspired model is related to the biological model *via the metalevel abstraction*, rather than by a direct naive mapping.

There is nothing in the conceptual framework [16] that explicitly corresponds to the CoSMoS results model. The purpose of the results model in simulation experiments is to compare the simulation with the real-world domain being investigated. Here the real-world domain (biological system) is not being investigated; it is being used as inspiration for the algorithm development. The domain model and platform model might be very far removed from each other. However, the reason that the biological system is being used as inspiration is that it has (usually emergent) properties that are wanted in the algorithm, for example, robustness. The results model could therefore be used to capture the algorithm's properties, and to validate that the algorithm as implemented has indeed captured these desired bio-inspired properties (as defined at some suitable level of abstraction, in the metamodel). Additionally, an

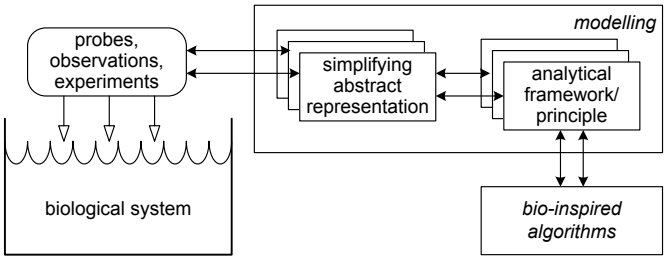


Fig. 5. A conceptual framework for bio-inspired algorithm design [16, fig.1].

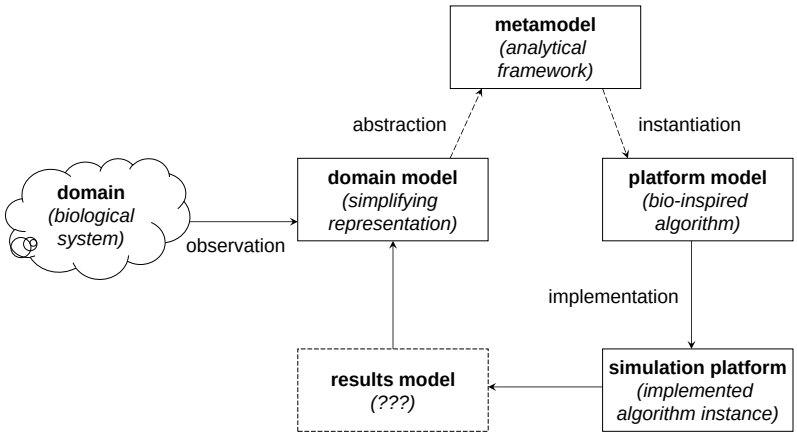


Fig. 6. The conceptual framework in terms of CoSMoS process models and metamodels. The framework components fit into the CoSMoS process. There is nothing in the framework explicitly corresponding to the CoSMoS results model.

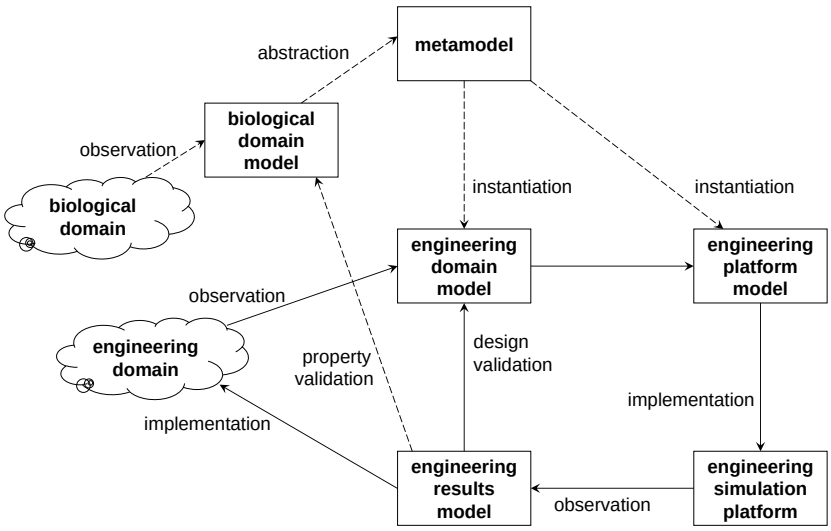


Fig. 7. A bio-inspired engineered domain.

explicit focus on these properties helps guide the initial domain modelling and abstraction process.

The description of the conceptual framework [16] does mention validation, but it is only a point-to-point process between consecutive models, and is not explicit about the purpose or means of the validation. Here we have an explicit place and approach to include validation of algorithm properties within the process.

4.2 Two domains

Bio-inspired algorithms are often developed for a particular purpose, for example, to engineer a particular bio-inspired system. The CoSMoS process for bio-inspired algorithm development described in §4.1 can be combined with the base simulation process of §2, as shown in figure 7. For example, this could describe the process of engineering a bio-inspired swarm robotics system.

The lower five boxes in figure 7, those joined by solid lines, represent a small modification of the base CoSMoS process. The main difference is that the domain is engineered, and so does not exist initially; the domain model is the engineering specification: the domain is engineered to respect the model (not the other way round, as when modelling a natural

domain). The 'design validation' demonstrates whether the simulation is an adequate simulation of the engineered domain.

The upper boxes represent the biological inspiration, as seen in §4.1. The 'property validation' demonstrates whether the engineered domain (via its capture in simulation) exhibits the desired biological properties.

This approach is not specific to bio-inspired engineering. It can cover any case where there is some 'inspiration domain' being exploited to help engineer a 'solution domain'. What is important is to understand what information is coming from what domain, and how the domains are linked, not directly, but through abstract models and metamodels.

We are currently applying this approach to developing a swarm robot system to investigate Simon's Loose Horizontal Coupling hypothesis [9, 14].

5 Summary and Conclusions

The CoSMoS process emphasises the building of a range of models, satisfying different purposes. In particular, the hypothesised properties (for example, emergent properties) are captured in the domain model, where they can be analysed, but are not present in the platform model, ensuring that they are not explicitly implemented. The results model allows the platform simulation outputs to be analysed and compared to the domain model.

Here we have introduced the use of an overarching metamodel in the CoSMoS process, to ensure that the various models are expressed in a common language, and to define the relationship between them.

We have shown an example of how the CoSMoS model and metamodel approach can be used to analyse pre-existing development frameworks, and to identify missing components that have a valuable function. We have also sketched how the process can be applied to simulations of bio-inspired engineered domains, in addition to simulation of pre-existing natural domains.

Acknowledgements

This work is part of the CoSMoS project, funded by EPSRC grants EP/E053505/1 and EP/E049419/1, and a Microsoft Research Europe PhD studentship. We thank the anonymous referees for helpful suggestions.

References

[1] Paul S. Andrews, Fiona A. C. Polack, Adam T. Sampson, Susan Stepney, and Jon Timmis. The CoSMoS process, version 0.1: A process for the modelling and simulation of complex systems. Technical Report YCS-2010-453, Department of Computer Science, University of York, March 2010.

[2] Anton Jakob Flügge, Jon Timmis, Paul Andrews, John Moore, and Paul Kaye. Modelling and simulation of granuloma formation in visceral leishmaniasis. In *CEC 2009*, pages 3052–3059. IEEE Press, 2009.

[3] Philip Garnett, Susan Stepney, Francesca Day, and Ottoline Leyser. Using the CoSMoS process to enhance an executable model of auxin transport canalisation. In Stepney et al. [17], pages 9–32.

[4] Philip Garnett, Susan Stepney, and Ottoline Leyser. Towards an executable model of auxin transport canalisation. In Stepney et al. [15], pages 63–91.

[5] Teodor Ghetiu, Robert D. Alexander, Paul S. Andrews, Fiona A. C. Polack, and James Bown. Equivalence arguments for complex systems simulations - a case-study. In Stepney et al. [18], pages 101–140.

[6] Teodor Ghetiu, Fiona A. C. Polack, and James L. Bown. Argument-driven validation of computer simulations – a necessity rather than an option. In *VALID 2010:*, pages 1–4. IEEE Press, 2010.

[7] Paolo Giudici. *Applied Data Mining: Statistical Methods for Business and Industry*. Wiley, 2003.

[8] Anneke Kleppe, Jos Warmer, and Wim Bast. *MDA Explained: the Model Driven Architecture: practice and promise*. Addison-Wesley, 2003.

[9] Jennifer Owen, Susan Stepney, Jon Timmis, and Alan Winfield. Exploiting loose horizontal coupling in evolutionary swarm robotics. In *ANTS 2010, Brussels, Belgium, September 2010*, volume 6234 of *LNCS*, pages 432–439. Springer, 2010.

[10] Fiona A. C. Polack. Arguing validation of simulations in science. In Stepney et al. [17], pages 51–74.

[11] Fiona A. C. Polack, Paul S. Andrews, Teodor Ghetiu, Mark Read, Susan Stepney, Jon Timmis, and Adam T. Sampson. Reflections on the simulation of complex systems for science. In *ICECCS 2010*, pages 276–285. IEEE Press, 2010.

[12] Fiona A. C. Polack, Paul S. Andrews, and Adam T. Sampson. The engineering of concurrent simulations of complex systems. In *CEC 2009*, pages 217–224. IEEE Press, 2009.

[13] Mark Read, Paul S. Andrews, Jon Timmis, and Vipin Kumar. A domain model of experimental autoimmune encephalomyelitis. In Stepney et al. [18], pages 9–44.

[14] Herbert A. Simon. The organization of complex systems. In Howard H. Pattee, editor, *Hierarchy Theory*, pages 1–27. George Braziller, 1973.

[15] Susan Stepney, Fiona Polack, and Peter Welch, editors. *Proceedings of the 2008 Workshop on Complex Systems Modelling and Simulation*. Luniver Press, 2008.

[16] Susan Stepney, Robert E. Smith, Jon Timmis, Andy M. Tyrrell, Mark J. Neal, and Andrew N. W. Hone. Conceptual frameworks for artificial immune systems. *International Journal of Unconventional Computing*, 1(3):315–338, July 2005.

[17] Susan Stepney, Peter H. Welch, Paul S. Andrews, and Adam T. Sampson, editors. *Proceedings of the 2010 Workshop on Complex Systems Modelling and Simulation*. Luniver Press, 2010.

[18] Susan Stepney, Peter H. Welch, Paul S. Andrews, and Jon Timmis, editors. *Proceedings of the 2009 Workshop on Complex Systems Modelling and Simulation*. Luniver Press, 2009.

Agent-based modelling of electrical load at household level

Jose Evora[1], Enrique Kremers[2], Susana Morales Cueva[2], Mario Hernandez[1], Jose Juan Hernandez[1], and Pablo Viejo[2]

[1] SIANI, University of Las Palmas de Gran Canaria, Las Palmas, Spain,
jose.evora@siani.es
[2] EIFER, European Institute for Energy Research (KIT & EDF), Karlsruhe, Germany
enrique.kremers@eifer.org

Abstract. Regarding electrical systems as complex systems offers new approaches for analysing, modelling and simulating those systems. Using software engineering techniques like Model Driven Engineering, a disaggregated model for household electricity demand is created. The Tafat framework for simulating complex energy systems is presented, including the concepts of the metamodel, models and behaviours. A first case study simulating the load curve of 1000 households composed of five different social groups is discussed and compared with an aggregated curve. The model is able to represent the load curve of a sample of households using a bottom-up approach.

1 Introduction

During recent years, an increasing interest in knowing more about electrical demand at low levels of the power grid can be observed. This is mainly due to the process of change the electrical system are undergoing, motivated by causes such as the introduction of renewable energy sources (RES) as well as distributed generation (DG). These changes require a better knowledge of the load curves at a distributed level, which involves different factors such as electrical equipment, user behaviour, environmental conditions, etc. that have a potential impact on the consumption. Aggregated data at substation level is no longer accurate enough to describe the consumption processes at the level of the distribution grid.

Modelling the consumption locally can help us to better understand the composition of aggregated load curves (also represented by load profiles) and analyse how local measures can have an effect on the global curve. Current aggregated models do not provide the possibility to model

local measures on the grid, as for example punctual actions taken by individual consumers, like switching on a cooking stove or an oven. They can only be included by modelling their aggregated effect and integrating it at aggregated level, due to their resolution. When considering the proposed changes in the electricity system, for planning and control activities at distribution level, these curves are not suitable, as they only fit for a large number of consumers because they describe the average use of energy over time [12, 4].

Some approaches already exist found which considered these facts and implement a demand modelling at lower scale. In [10] a simplified demand model for domestic lightning is presented that provides estimating the aggregated demand or even the distributed loading for groups of households. In [4] a multi-use bottom-up domestic consumption modelling tool was developed and verified. [14] gathered and analysed high resolution domestic electrical data and found that logging at intervals of few minutes is necessary to capture the fine detail of load patterns for evaluating on-site generation. In [11], a high resolution stochastic approach, using heterogeneous Markov chains is chosen to model domestic electricity demand.

2 Electrical systems

The electric power system is composed of different electrical components that allow for the production, transmission and consumption of electric power. Production or generation of electric power is the process of converting energy in other forms (chemical, mechanical, nuclear, etc.) into electrical energy. For the transmission, different kinds of electrical networks are used. Generally, they are classified by their nominal voltage at each level. Long range transmissions over hundreds of kilometres are performed at high voltages of several hundreds of kilovolts (kV). These networks are called transmission systems. Transformer stations (substations), can reduce the voltage at a given point in order to feed electricity to the so called distribution system. The distribution system carries the electricity to the final consumers. These networks operate at medium voltage levels, usually between 1-50 kV. In a final stage, distribution transformers can convert from medium voltage into low voltage (less than 1 kV) which is the typical voltage level find at residential or tertiary customers. Some specific customers (such as industries) may have direct connections at medium voltage level, too.

In a *classical* energy system, generation is injected at high or medium voltage level and consumed in the distribution system. Following the trend of introduction of renewable energy sources and distributed gener-

ation, injections of energy at almost all levels of the system are possible. Also, and in order to better match the fluctuating production introduced to the system, Demand Side Management (DSM) mechanisms, which allow to manage the consumption are being developed. The electricity system can be seen as a complex system, being composed of a large number of interacting entities. The reproduction of the behaviour of the system is therefore not possible by modelling only individual objects or the system in a monolithic way. In this paper, we use a disaggregated method to model a part of the electricity system. A detailed model of low-voltage demand is constructed for simulating the load curve of individual households. This includes the loads of each household, representing the final end consumers of this electrical system. Simulating a large number of entities, the individual consumptions can be aggregated and a representation of a load curve at e.g. substation level is created.

2.1 Complexity and intelligence in electrical systems

Classically, the electrical system management has been done based on the aggregated consumption data at a global level. The nature of this management is has been centralised since the system was considered as an indivisible unit. When new devices that apply demand side management strategies are introduced in real electrical systems, new management conceptions must be considered. So, the management of the future electrical system must overcome the restrictions that are introduced now and start analysing the electrical system as a distributed system.

This conception of the electrical system shows analogies with other living complex systems, such as ants or bees colonies, in which there are several agents that are taking decisions and acting locally. Actions that each agent are executing locally are aggregated, and these actions can lead to emergent phenomena.

From our perspective, in the electrical systems, people living in the household are agents [13]. They can be considered as intelligent [3] since they are self interested units and exhibit an adaptive behaviour to their environment. These agents are able to take decisions and coordinate their actions with other agents. The main difference with the study of living complex systems is that in the electrical system, we are not interested in the study of the emergent behaviour starting from the local actions, but the modification of local behaviours to get the desired emergent behaviour [9]. However, in a second point of view, a complex system simulation model can serve to observe unwanted emergent phenomena and study these effects on the system. An approach to analyse complex systems from this point of view, at the very bottom, can be seen in [6].

2.2 Analysing electrical systems

From an analytical point of view, in electrical systems, we can classify objects according to their behaviour in three categories: first entities that can be described in term of their physics such as a building that exhibits a thermodynamical behaviour, second entities that can be described with a mechanistic model such as a washing machine and third agents that can be described with a intentional model such as people living in a household or smart meters.

This behaviour separation match perfectly with the differentiation proposed in [2]. In this work, the author proposes the following three behaviour categories from a observer point of view:

- **Physics Stance:** at the level of physics and chemistry. It is concerned with things such as mass, energy, temperature, velocity, and chemical composition.
- **Design Stance:** at the level of biology and engineering. It is concerned with things such as the function of a living system or the design of a system.
- **Intentional Stance:** at the level of software and minds. It is concerned with things such as belief, thinking and intention.

3 Simulation of electrical systems

Like other complex systems, electrical system simulations involve the behaviour execution of the many elements that exists in a real grid.

The process starts by modelling the electrical system using Tafat. Tafat is a framework for building simulation through Model Driven Engineering which is currently under development by SIANI and EIFER. It is currently implemented in JAVA, but designed to be language-independent. Tafat uses an object oriented and agent-based approach. In this framework, each element of the system is represented including a behaviour that explains how it changes over time. Then, a simulation is run by executing all the behaviours concurrently. During the simulation, behaviours can modify the attributes of these entities. One entity can have different kinds of behaviours, that are stored in a repository. The separation of structural and behavioural elements (entities, defined in the metamodel; and behaviours, stored in a repository) should be underlined here.

However, since the electrical system is locally and massively affected by the human interventions, it is also required to model this behaviour along with other entities that belong to the electrical system architecture. Representing the human behaviour means that it is required to define

what people do during a time period, which electrical devices they have and how they use them.

Until now, the human behaviour has not been analysed and electrical companies only have historic aggregated data of the consumption at global level. From now, it is quite important to have data at the household level to be able to model households and simulate the behaviour of people inside. So, the main challenge in modelling electrical systems is to describe all the electrical appliances in the households and the behaviour profiles whose actions generate electrical consumption.

3.1 Software engineering approach

It is needed to approach the software engineering of the simulator with a methodology that supports the modelling of a large number of entities. The goal is to make the simulator development and maintenance easier.

The approach that has been taken is based on a software engineering paradigm called Model Driven Engineering (MDE) in which systems are developed at an abstraction level close to the problem domain. In this approach, models are the artifacts that drive the development process. From an evolutionary point of view, development methods have been trying to get more abstractions in order to reduce the gap between the programming semantic and domain semantic.

MDE is the result of the combination of technologies such as Domain Specific modelling Languages (DSML) and engines that analyses models for synthetising software artifacts. This paradigm allows to improve productivity and flexibility at developing a simulator, since the application is developed and modified just by changing the model [8].

3.2 Metamodel and models

To model the electrical system, a metamodel for this domain has been developed. The metamodel is a formal representation that supports the description of an electrical system with a semantic that is closer to the domain expert. The metamodel can be understood as a representation language for modelling electrical systems.

This metamodel comprises the classes of entities that have been identified in the real world. So, the metamodel defines what kind of elements can be represented in the model and provides a standardised way of representing elements. Classes are represented in a model by means of context, features, variables and behaviour (Figure 1). Context defines where elements can be placed in the scene. Features are defined as static data of an element; while variables are defined as dynamic data. The behaviour is the logic that updates the dynamic data.

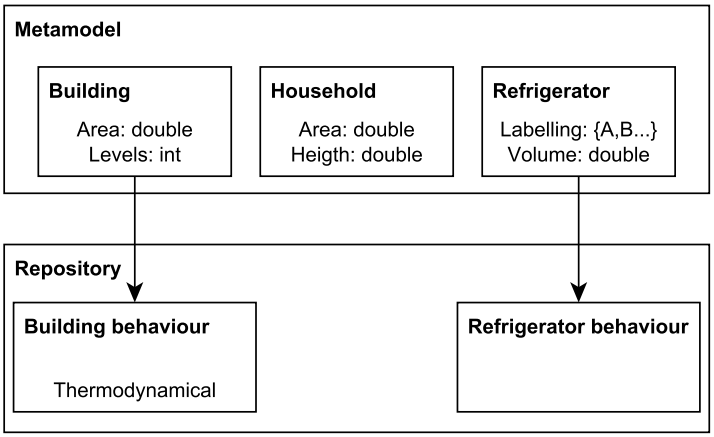


Fig. 1. Structure of the metamodel where the element definitions are exposed.

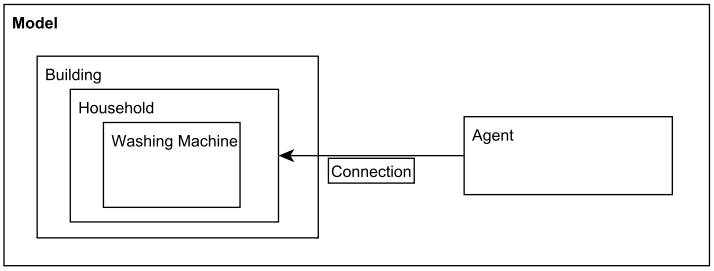


Fig. 2. Structure of the model shown by an example including behaviours

In the metamodel, we distinguish different types of model elements: entities, static objects that belong to the scenario (e.g. refrigerator); agents, intentional objects that interact with entities and communicate with other agents (e.g. person); connections that define relations between entities or agents (Figure 2).

A model is a representation of a specific electrical system using the metamodel classes. Elements that are going to be simulated are instantiated from the metamodel in the model. Both, models and the metamodel, are represented using XML.

3.3 Programming behaviours

To simulate the electrical system, all the metamodel classes have to include behaviours that must be programmed.

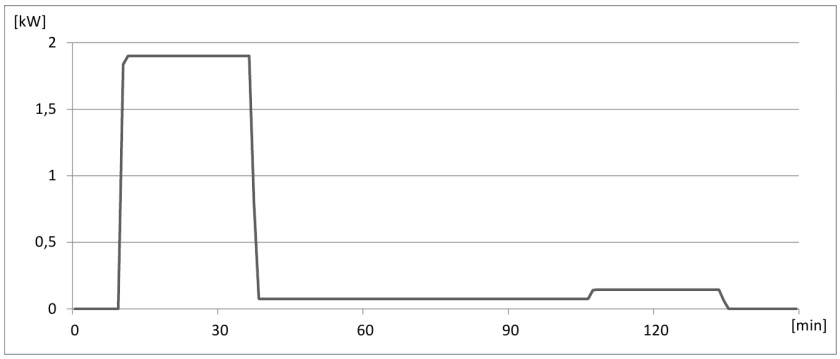


Fig. 3. Simulated load curve of a washing machine

A separation between the different types of behaviours must be done according to the discussion exposed in the section 2.2. In this section, an example of each kind of behaviour will be exposed.

Environmental behaviour Environmental behaviours represent the change over time of some environmental variables. Environmental variables are normally common to a group of entities or devices and describe the surroundings or "outdoor". These can be for example solar radiation models, which represent the insolation and can be used for calculation of the thermal gains of a building, or further for energy production (photovoltaics, solarthermal use, etc.). These behaviours do not directly change the attributes of a device or agents, but rather allow some interactions in an indirect way (e.g. through heat exchanges, etc.).

Device behaviour Most of the electrical devices used in a household are major appliances such as washing machines, refrigerators, etc. There are also some other, smaller appliances, such as CD players, TVs, HiFi Audio equipment, etc. Usually, the major appliances cause a larger part of the electrical consumption. In order to recreate the individual load curves, EIFER (European Institute for Energy Research, a common research institute by KIT and EDF) has developed individual models for the behaviours of electrical appliances, which were integrated into Tafat. Simplified technical models are used, which take into consideration different technical parameters of a specific appliance. So, for example, the load curve a TV will be characterised by the size and technology (CRT, LCD, Plasma, etc.). Major appliances also are modelled using the EU Energy Label as an input parameter, which is an indicator for the energy consumption of a device and is compulsory for appliance sold in the

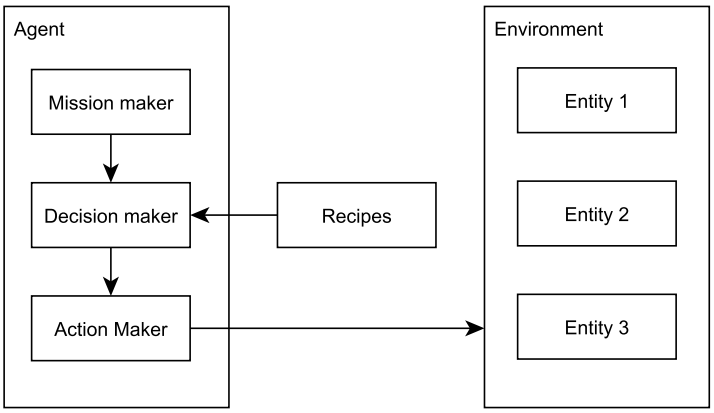


Fig. 4. Agent architecture composed by a mission maker (intentions), decision maker (recipe selector) and action maker (recipe executor)

EU. Different releases for the behaviours of the electrical devices were created. Using this modular approach, a behaviour of a single device can be *exchanged* in a simple way. The different releases include simplified technical models with varying degrees of accuracy, thus allowing for an optimisation of execution time vs. accuracy of the model. In Figure 3 an example of the load curve generated by the behaviour of a washing machine can be seen. This load curve is created by a simplified technical model of this appliance.

Social behaviour A flexible architecture is proposed to carry out a social behaviour. As described in section 2.2, intentional stances are the most complex behaviours. For this reason, the architecture must be flexible to allow a range of behaviours from simple behaviour based on a list of tasks to a complex behaviour implemented as a neural network.

The mission maker is the intention launcher, the decision maker is in charge of choosing a recipe to accomplish the mission launched and the action maker is the executor of the recipe. The recipe is a list of actions that executes to accomplish a mission. With this architecture, a simple behaviour can be developed by creating a big recipe in which all the tasks are described and having a mission and decision maker very simple. Otherwise, in a complex social behaviour, the task can be launched by the mission maker according to several parameters of its own agent or the environment, having a hard process to choose a recipe in the decision maker, but easier recipes that only describe how to arrange a task as, for example eat.

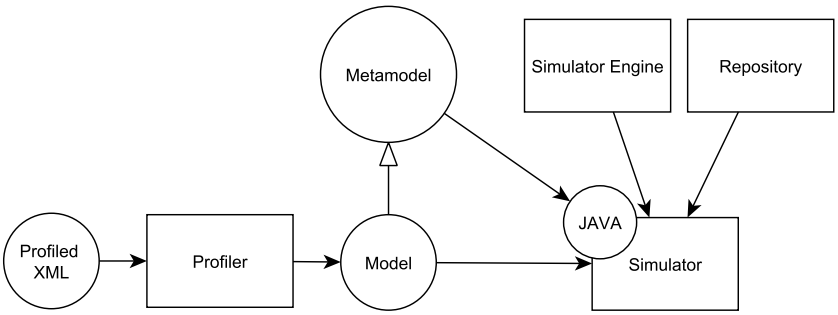


Fig. 5. Tafat architecture

3.4 Simulation

The main problem for developing good models that represents accurately a place (town) as the lack of data. Often, it is quite difficult to gather the needed data.

Ideally, models should be built using real data, since the simulation will help to understand what happens in the electrical system. However, since data is not available at all, a model approximation is done. A tool called profiler has been developed (Figure 5), which helps to carry out this task. Using a high-level description of a place (for example, amount of buildings and population), Profiler automatically generates an electrical system model that can be simulated directly. The profiler is part of Tafat, a MDE platform that supports the development of simulations.

Electrical models can be created to represent the load accurately but light-weight enough for use in large scale simulations, and handle demand side management mechanisms through the use of an agent-based approach.

4 Case Study

4.1 Modeled scene

The case study proposed is an analysis of the electrical load of households according to social group characteristics. The following five different social groups have been taken into account to develop the case study:

1. junior single,
2. senior single,
3. junior couple,

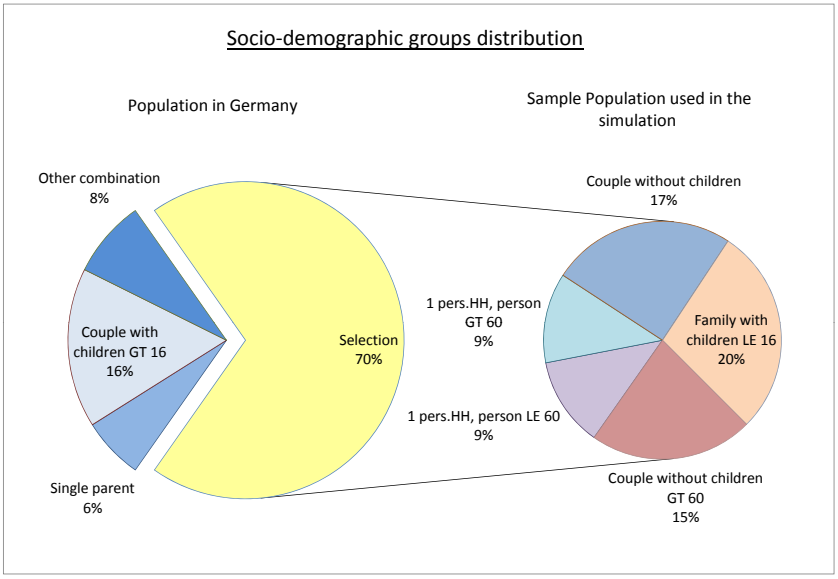


Fig. 6. Socio-demographic groups used in the case study. Source: SOEP.

4. senior couple and
5. family with children.

These groups were taken as a part which represents about 70% of the population of Germany, being the most numerous groups identified by the German Socio-Economic Panel Study (SOEP). The constitution of this sample is shown in Figure 6.

A survey[3] has provided the information about the timetables and appliance usage of a household according to the social group. Based on this information the agent behaviour which is related to the household is implemented. The electrical usage behaviour data employed for this study was obtained through a local survey. Thus the gathered data from a small sampling was used as input parameter in the Tafat model. Hence, 20 different social behaviours (i.e. 20 different model recipes) are been used to simulated the five socio-demographic groups. Each of these recipes includes some randomness through the definition of intervals at

[3] The survey consisted in local interviews with around 20 persons in Karlsruhe, Germany in order to obtain data like usage times and durations of specific socio-demographic groups. It has to be noted that the survey is not representative but rather a sample of the user behaviours of those groups.

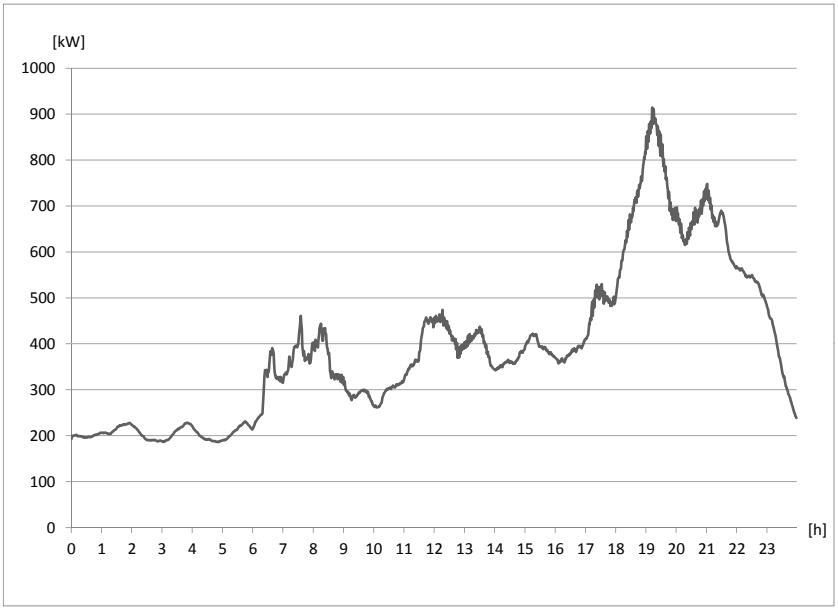


Fig. 7. Simulated load curve of 1000 households for one day

which devices are switched on or off. The exact time instant of time is obtained as a uniformly distributed random variable over this interval.

To arrange this study, an important utility of Tafat to build automatically a scene has been used. This utility uses statistical data from the SOEP and the survey to create a model scene in which the social groups are distributed in the households. Those households contain the electrical devices, and their amount, according to the social group.

4.2 Simulation and results

In the simulation, the device load curves within a household are generated. The curves were aggregated using an individual behaviour for each household taking into account some randomness (variation of the duration and use time in a defined range of the different electrical devices).

The simulation results show the load curve of a day of 1000 households with around 12 appliances each, composing thus an amount of approximately 12000 simulation elements. This type of simulation can provide the relevance of a determined power consumer in the household consumption as well as the influence of a specific type of consumer on

the global load. The number of 1000 entities was chosen, as it was determined that larger amounts did not change substantially the results and only increased the execution time of the simulation. This is probably due to the use of a limited number of recipes (taken from the number of surveyed persons). In this case, the execution time was around 20 minutes on a standard desktop PC for the simulation of a period of 24 hours.

The 1000 household sample is composed of a distribution of the different social groups according to real statistical data, in order to obtain a sample of households that is as close as possible to reality. In Figure 7, the simulated load curve for one day can be seen. The simulation is run in a high time resolution, being the time step one minute. This allows observing effects which are neglected in simulations at lower resolutions, provided by 15 minute or hourly models. Some sharp peaks can be observed, which are caused by the use of high power consuming devices in the household. A general trend to use more power during the day is clearly visible. During the night, the base load (devices that are constantly running, such as refrigerators and other permanent loads) cause a consumption that is only around a third part of the daily peak load. The configuration of the simulation can be seen in Figure 8.

In Figure 9 the curve is compared to a standard load profile of Germany for a winter weekday (according to [1] and made public by [7]) for household demand. The profile is provided in a normalised form in order to be weighted with a given number of energy. In this case, and for comparison purposes, the profile was weighted with the same amount of energy as the simulation curve, i.e. that the daily consumed energy [kWh] is the same in both cases. The load profile is a smooth curve, representing an aggregated load behaviour at high levels of the electricity system for large number of consumers. The are the result of a statistical analysis based on representative samples from different consumer groups [5].

The simulated curve represents the total power consumed by a sample of 1000 households, modelled individually and with an autonomous behaviour for each of them. The curve has been adapted by averaging periods of 15 minutes in order to match the same time granularity as given by the standard load profile. The curve is more peak shaped, which is probably due to the relatively small amount of households (in comparison to the statistical samples taken to obtain a profile, which are representative) and the reduced number of behaviours (in total, only 20 different behaviours have been used). Furthermore, only 70% of the household population is modelled, neglecting other social groups which may change the curve.

Entity	Attributes	Components	Behavior
Building		Household: n=1	-
Household	sg = montecarlo[0] Households are classified into one of the 5 social groups (sg) using montecarlo method area = $f_{area}(sg)$ Area is variable depending on the social group	PowerConsumers CookingStove n=4 Microwave n=1 Oven: n =1 Dishwasher: n = 1 Refrigerator: n = 1 AudioHifi: n = 1 TV: n = f_{tv}(sg) Ligthing: n = 1 Washing Machine: n = 1 Computer: n=$f_{computer}$(sg) Waterboiler: n = 1 Hairdryer: n = 1 Vacuum: n = 1 Iron: n = 1 Console: n = $f_{console}$(sg) PowerBus	Sociological Behavior Each social group has 4 different recipes. A recipe is randomly selected. The recipe defines how the power consumers **mode** is changing
PowerConsumer	mode power		DeviceBehavior Power is calculated depending on the device mode and on its technical characteristics
PowerBus	power		DeviceBehavior power is calculated by aggregating the consumption of Household PowerConsumers

Fig. 8. Configuration of the entities used in the simulation.

Even though the selected samples in the survey are not representative for all Germany nor society as a whole (only five social groups were used), the general trend of both curves are similar. Three peaks can be observed, which are closely synchronised in time and correspond to morning, noon and evening peaks. These peaks are correlated with a large and concurrent usage of high power devices, such as cooking plates, ovens, microwaves, etc. due to alimentation habits, as well as lightning use in the evening hours. The morning and noon peaks are lower in the simulation than in the profile, whereas the evening peak is higher. The timely synchronisation of the ramps of the peaks matches quite well; this indicates that the activities (having breakfast, lunch, returning home, etc.) were modelled according to the average German user behaviour. Even though, some differences can be observed at the evening drop (21-23h), as well as a small second peak that cannot be found in the profile.

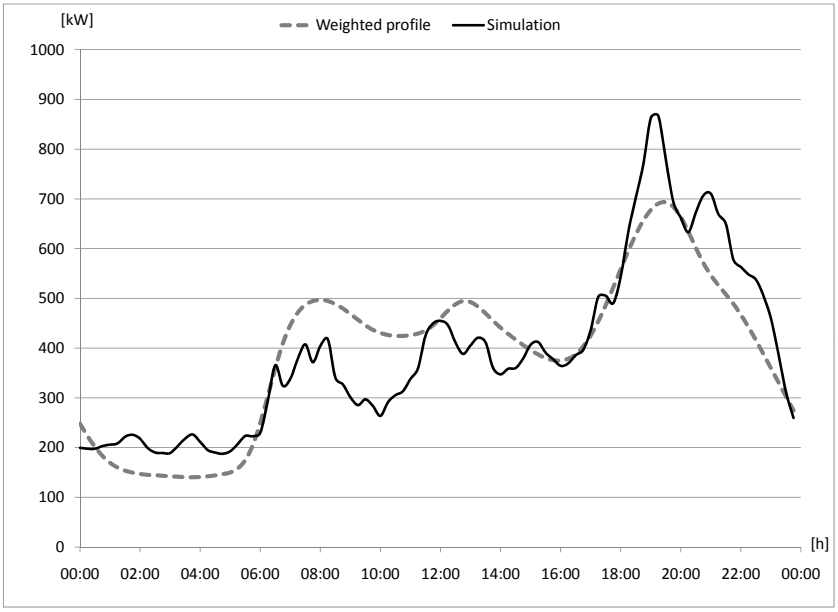


Fig. 9. Simulated load curve vs. standardised residential load profile

4.3 Discussion

Even using a relatively small sample of households and reduced number of behaviours, a curve that represents the major characteristics (peaks and troughs, as well as their timely synchronisation) is generated. Concerning the differences in the height of the peaks, the model could be reviewed in order to check the individual power curves of each electrical device. This seems to be quite hard as almost no data is available for such a validation (at a representative sample). However, the differences could also be related to the use of only 70% of the socio-demographic population share. Further, behaviour itself is another factor to consider, as a largely simplified and almost static model was used. Some specific characteristics of the model create peaks, which could be explained by a rather homogeneous behaviour of the groups. For example the second evening peak (around 21:30h) is possibly caused by some activities (watching TV, other evening activities) which start and end at similar times, because of the relatively small sample. Using data from a more representative survey or a more stochastic-based social behaviour, this could be avoided, though.

5 Conclusions and outlook

In this paper, a vision of the electrical system as a complex system has been introduced. This is necessary as the future grid behaviour becomes more distributed. Furthermore, the simulation of the electrical system at a household level as a complex system has been addressed.

The case study demonstrates that a bottom-up simulation of the residential consumption using an agent-based approach has been successful since the result curves show similarities to aggregated load curves. A comparison with a national aggregated profile shows similarities in the main characteristics of the curve. Moreover, due to the high resolution of the model, a large number of parameters (individual appliances types and models, socio-technical behaviours, etc.) is available for variation.

From now, the simulation that has been developed will allow us to experiment and study new algorithm and strategies for electrical system management. New scenarios, new problems and new challenges will arise in the near future with the introduction of renewables and a distributed production in the electrical system. Simulation is particularly necessary to design a new management approach.

The simulation will allow us with further work to study the integration of demand side management strategies. Strategies, such as adaptive or reactive technologies, incentives or campaigns, can be addressed for studying their impact at load curve level.

However, social behaviours need to be validated and improved. This validation is necessary for studying the emergent behaviours and for identifying the local actions of agents which provokes the desired emergent behaviour. Moreover, social behaviours must be improved in order to introduce a higher degree of heterogeneity to the models. Due to the high resolution of the household model, individual actions such as changing specific parameters on a device can be performed.

Furthermore, the model developed could be expanded in order to simulate not only the demand side, by including distributed generation or other injections to the grid (like storage), which could interact with the existing elements. This is already contemplated within Tafat, and would allow a disaggregated analysis of offer-demand balance and the possibility to estimate the impact of those measures at a system level.

References

[1] Bundesverband der Energie- und Wasserwirtschaft. BDEW - Standardlastprofile (SLP), 2011.

[2] D. Dennett. *The Intentional Stance.* M.I.T. Press, 1987.

[3] A. Monti, F. Ponci, A. Benigni, and J. Liu. Distributed intelligence for smart grid control. In *Nonsinusoidal Currents and Compensation (ISNCC), 2010 International School on*, pages 46–58, 2010.

[4] Jukka V. Paatero. *Computational Studies on Variable Distributed Energy Systems*. Phd thesis, Helsinki University of Technology, 2009.

[5] P. Palensky, F. Kupzog, A. A. Zaidi, and Zhou Kai. Modeling domestic housing loads for demand response. In *Industrial Electronics, 2008. IECON 2008. 34th Annual Conference of IEEE*, pages 2742–2747, 2008.

[6] Fiona A. C. Polack, Tim Hoverd, Adam T. Sampson, Susan Stepney, and Jon Timmis. Complex systems models: Engineering simulations. In *ALife XI, Winchester, UK, August 2008*, pages 482–489. MIT Press, 2008.

[7] RWE Rhein-Ruhr Verteilnetz. Lastprofile, 2011.

[8] Douglas C. Schmidt. Guest editor's introduction: Model-Driven engineering. *Computer*, 39:25–31, February 2006.

[9] S. Stepney, F. A. C. Polack, and H. R. Turner. Engineering emergence. In *Engineering of Complex Computer Systems, 2006. ICECCS 2006. 11th IEEE International Conference on*, page 9 pp., 2006.

[10] Melody Stokes, Mark Rylatt, and Kevin Lomas. A simple model of domestic lighting demand. *Energy and Buildings*, 36(2):103–116, 2004.

[11] Joakim Widén and Ewa Wäckelgard. A high-resolution stochastic model of domestic activity patterns and electricity demand. *Applied Energy*, 87(6):1880–1892, 2009.

[12] H. Lee Willis and Walter G. Scott. *Distributed power generation: planning and evaluation*. Marcel Dekker, New York, 2000.

[13] Michael J. Wooldridge. *An Introduction to Multiagent Systems*. John Wiley, Chichester, West Sussex, United Kingdom, 2002.

[14] Andrew Wright and Steven Firth. The nature of domestic electricity-loads and effects of time averaging on statistics and on-site generation calculations. *Applied Energy*, 84(4):389–403, 2007.

A computational technique to scale mathematical models towards complex heterogeneous systems

C. H. McEwan[1,2], H. Bersini[3], D. Klatzmann[1], V. Thomas-Vaslin[1], and A. Six[1]

[1] UPMC Univ Paris 06, UMR7211, I2D3 Integrative Immunology: Differentiation, Diversity, Dynamics; 83 Bd de l'Hôpital F-75013 Paris, France
[2] CNRS UMR7606, LIP-6, Agents Cognitifs et Apprentissage Symbolique Automatique; 4 Place Jussieu 75005 Paris, France
[3] IRIDIA Universit Libre de Bruxelles, 50 Av. F. Roosevelt, Brussels, Belgium

Abstract. In this paper, we report progress on applying techniques traditionally used by computer scientists for specifying finite state machines, to concisely express complex mathematical models. These techniques complement existing graphical methods in Systems Biology by allowing a systems approach to be taken at a coarser granularity than biochemical reactions – where parallel, multi-level interactions within and between systems must be documented, communicated and simulated.

1 Introduction

Modelling of complex systems is a persistent problem in science. In principle, modelling is the means by which empirical observations can be explained and predicted by mechanistic reasoning. In practice, modelling is fraught with many methodological and pragmatic trade-offs. One particularly divisive trade-off has traditionally been the commitment to either population-based Ordinary Differential Equations (ODE) or individual-based Agent-Based Modelling (ABM). The former has a proud history across the sciences, with a strong theoretical foundation and well developed methods. The central idea is the aggregation of individuals into homogeneous compartments with mean behaviour; which is both the strength and weakness of the approach. It might be argued that, for typical "complex systems", homogeneity is not a defining feature [18]. In contrast, ABM support very fine-grained heterogeneity by composing

arbitrary agent behavioural rules. However, this approach has little theoretical or methodological foundation [14]. For non-trivial models, ABM is computationally prohibitive, rendering impractical the many executions necessary for parameter fitting, sensitivity analysis and statistical confidence. Furthermore, non-negligible technical details of either methods tend to be opaque to experimentalists, whom these models claim to afford insight. A complementary problem exists, ensuring the fidelity of interpretation of an experimentalist's phenomenological description.

Our proposed attack on these problems utilises techniques developed by computer scientists for specifying Finite State Machines (FSM). FSM are the *de facto* engineering approach to developing many real-time safety critical systems, such as those in the avionics industry [8] and these systems tend to be significantly more complex than typical mathematical models in science. This paper will demonstrate that the techniques that allow FSM to scale to such complexity can be applied to mathematical modelling and improve upon existing graphical modelling frameworks. This benefits both the modeller and collaborators: producing models that approach the intuitiveness and heterogeneity of individual-based models, but support the formalism and efficiency of population-based models.

The remainder of the paper is outlined as follows. In Sect. 2 we briefly review the work we intend to draw upon. In Sect. 3 we introduce a hypothetical modelling effort to motivate and serve as an illustration for later sections. In Sect. 4 we formalise our ideas, demonstrating their application in Sect. 5. We conclude in Sect. 6 with some discussion and future work.

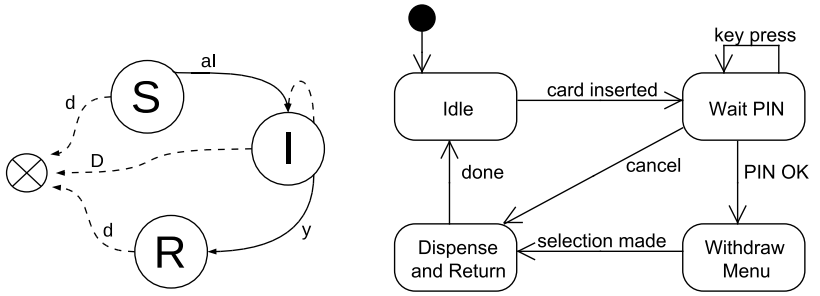
2 Background and related work

Classically, an ODE model is presented as an n-dimensional state vector $x(t)$, representing the n quantities of interest at time t, and a system of n coupled differential equations of the general form

$$\frac{dx_i}{dt} = f_i(x,t) = \sum_{j=1}^{n} f_{ij}(x,t) - \sum_{j=1}^{n} f_{ji}(x,t), \tag{1}$$

where each f_{ij} quantifies the flow of "mass" into x_i from x_j. For maximal generality, each f_{ij} is presented as an arbitrary function of the current state x. In practice, f_{ij} may be a function of x_j, $x_j x_k$ for $k \in [1,n]$, or a constant. Typically, many $f_{ij} = 0$ (coupling is sparse) and none depend on time t (the system is autonomous). What follows does not depend on these conventions.

This deliberately over-general presentation makes the graphical form of an ODE readily apparent (see Fig. 1(a)): each variable x_i can be



(a) ODE flow diagram (b) Finite state machine

Fig. 1. ODE and FSM models as directed graphs. **(a)** The classic "susceptible, infected, recovered" model from epidemiology, parameterised by rates of flow between population compartments. The ODE can be read off from this diagram, e.g. $\frac{dI}{dt} = aIS - DI - yI$. **(b)** A simple finite-state representation of a cash machine. Transitions occur in response to discrete external events generated as the customer interacts with the machine.

represented as node and each f_{ij} as a directed edge. Such graphical approaches are sometimes used for pedagogical purposes, under the the moniker of *flow charts* [15]. Graphical modelling is also popular in Systems Biology (see e.g. [11]) although the approach we introduce here will be more widely applicable than chemical kinetics.

Although conceptually simple, it is perhaps worth noting that the graphical representation in Fig. 1(a) is *both* more readily accessible to those untrained in mathematical modelling *and* more concise than the corresponding mathematical expression and its numerical implementation. For example, each term f_{ij} in Eq. (1) appears at least twice, in the incoming summation of $f_i(x)$ and in the outgoing subtraction of $f_j(x)$. In models less abstract than Eq. (1) this can make for an error-prone developmental process. In contrast, simulation of Fig. 1(a) essentially amounts to enumerating each directed edge and transferring mass between it's end-points. This is more akin to *stoichiometric* "reaction channels" in chemical kinetics [4], of which Eq. (1) can be seen as a special case.

2.1 Finite state machines and statecharts

In contrast, Finite State Machines are a computational formalism that are explicitly based on directed graphs (see Fig. 1(b)). Here, each node represents a possible "state" of the machine; each edge represent the change of state that occurs in response to a discrete "event". There are

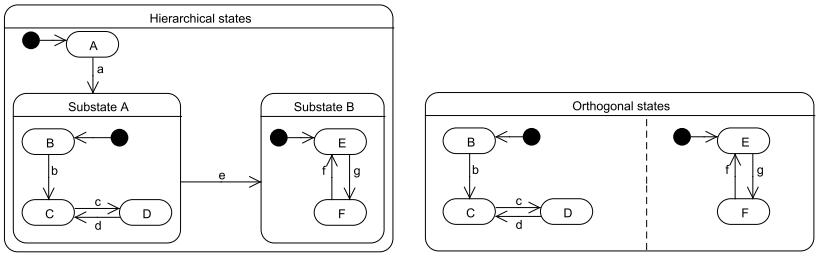
several flavours of this formalism, though the details are not important here. Although Fig. 1(b) is intuitive, as FSM become more elaborate the complexity of the graph structure rapidly grows beyond that of the behaviour it describes. In the worst case, introducing a state may result in a combinatorial increase in the elements of the graph.

To address this problem, Harel introduced statecharts [7], a graphical formalism for the unambiguous representation of complex finite state machines. The impetus for this contribution was concern not unlike that we see voiced today in biological modelling, e.g. a need for hierarchical, multi-level organisation with a controllable level of granularity; and representing systems composed of complex internal sub-systems that operate in parallel. Indeed, Harel is a vocal proponent of applying this formalism to modelling complex biological systems [3, 5, 19], albeit as an ABM paradigm. Others have incorporated state machine and ODE formalisms in so-called "hybrid models" [9]. Recently, statecharts were proposed as an aid in producing logical (i.e. qualitative) models [16]. In contrast to preceding work, we apply statecharts to quantitative, population-based modelling of continuous-deterministic and discrete-stochastic systems.

The most prominent contribution of statecharts was the introduction of *hierarchy* and *orthogonal parallelism* as organisational constructs. These allow the diagram to remain closer to the conceptual complexity of the model, while retaining the necessary expressive power to specify all possible execution paths of the model unambiguously. Informally, the motivation for both is thus:

Hierarchical States (see Fig 2(a)) aggregate simple states into compound states that are implicitly entered when one of the sub-states is entered. This compound state provides a graphical element where similar functionality, e.g. transitions, shared by all sub-states can be refactored to minimise diagram clutter. This hierarchical organisation also allows the modeller to "zoom" into specific levels of model detail. Similarly, a simple state may be expanded into a hierarchical state as more detail about the system becomes available.

Orthogonal States (see Fig. 2(b)) factorise the diagram horizontally – decoupling sub-systems that are "almost independent". Clearly, if these sub-systems were truly independent then they could also be modelled and simulated independently. The value of orthogonal states is that they provide a manner to specify just the deviation from independence, while diagrammatically representing the sub-systems *as if* they were independent. The orthogonal state is a minimal representation of something that would be graphically and organisationally unwieldy, though largely re-



(a) Hierarchical statechart (b) Orthogonal statechart

Fig. 2. The two fundamental ideas behind statecharts. **(a)** Hierarchical states representing different granularity of detail. A machine in states B, C or D is implicitly in $SubstateA$. Common elements can be refactored, minimising clutter, such as the transition e that is a valid transition from B, C or D. **(b)** Orthogonal states represent a machine that must be in *one of each* states separated by dashed lines, e.g. C and E. These constructs can be composed arbitrarily to express complex internal behaviours.

dundant due to the combinatorial nature of the underlying FSM. This will be demonstrated in further detail in Sect. 3.

There are of course other aspects to the statechart formalism. Some will not make the translation to mathematical models, others will be introduced as needed. The key point here is that although the interpretative semantics of the directed graphs may be quite different for ODE and FSM formalisms, the techniques we introduce below largely operate on the underlying graph structure, *not these semantics*. This combinatorial perspective allows us to resolve a difficult issue with developing and communicating complex mathematical models – the proliferation of variables, their evolution equations and the terms in these equations – that has already been solved in the context of FSM. In turn, complex system models become more scalable for both theorist and experimentalist collaborators.

3 Cellular processes: a motivating example

To avoid digressions that distract from the technique itself, we develop a toy model that can be easily followed and succinctly explained. We also take some liberties with the true phenomenon in order to demonstrate specific points. The reader should keep in mind that the techniques introduced are applicable to models of greater complexity than that shown.

The level of complexity we have in mind can be seen in efforts such as [1, 18].

3.1 The complexity of systemic models grows multiplicatively

Consider a hypothetical model of a lymphocyte that, after an initial period of "naivety", has its receptor bind to its ligand. This induces the cell to become an "effector", performing some function not relevant here. After a period, effector functionality wears off and this binding process can repeat indefinitely, but the cell is now upgraded to a "memory", rather than "naive", cell. Perhaps effector cells proliferate more heavily and memory cells have a lower death rate. Such models are typical in theoretical immunology and a common modelling step would be to examine if and how a population of such cells reaches a homeostatic configuration given initial conditions and appropriate parameter values for binding, maturation, proliferation and decay. A likely model might be

$$\frac{dN}{dt} = b + p_1 N - (d_1 + a(N, M, L))N \tag{2}$$

$$\frac{dE}{dt} = p_2 E + a(N, M, L)N + a(M, N, L)M - (d_2 + y)E \tag{3}$$

$$\frac{dM}{dt} = p_1 M + yE - (d_2 + a(M, N, L))M \tag{4}$$

where N, E, and M represent naive, effector and memory cell populations, respectively, and b, d_i, p_i and y are rates of birth, death, proliferation and maturation. The non-linear function a quantifies competition for and binding to ligand L, the details of which have no immediate bearing on our presentation. Now, lymphocytes spend their life between peripheral tissues in the body and the lymph nodes, where antigenic debris is drained and collected. Not only is there a greater concentration of ligand in the lymph node, but also the appropriate chemical environment for sustained proliferation. We need to expand our model to account for these facts, which requires replicating Eqs. (2)-(4) to represent dynamics inside and outside the lymph node, as well as flow into and out of the lymph node itself. Perhaps

$$\frac{dN}{dt} = b + p_1 N - (d_1 + j + a(N, M, L))N + kN' \tag{5}$$

$$\frac{dE}{dt} = p_2 E + a(N, M, L) + a(M, N, L) - (d_2 + j + y)E + kE' \tag{6}$$

$$\frac{dM}{dt} = p_1 M + yE - (d_2 + j + a(M, N, L))M + kM' \tag{7}$$

$$\frac{dN'}{dt} = p_1 N' - (d_1 + k + a(N', M', L'))N' + jN \tag{8}$$

$$\frac{dE'}{dt} = p_3 E' + a(N', M', L') + a(M', N', L') - (d_2 + k + y)E' + jE \tag{9}$$

$$\frac{dM'}{dt} = p_1 M' + yE' - (d_2 + k + a(M', N', L'))M' + jM \tag{10}$$

where primed variables represent lymph-node specific quantities, j and k represent flow into and out of the lymph node, respectively, and p_3 represents the increased proliferation rate due to the favourable environment.

We now introduce an experimental treatment where dividing cells can be induced to initiate apoptosis, or cell death (see e.g. [20]). To emphasise our point, at some cost to plausibility, we will assume that this treatment takes effect at any checkpoint during the cell-cycle. This simply provides a multi-level, finer granularity to the modelling task, while avoiding contrived realism. For a basic five-stage cell-cycle, this development would require replicating Eq. (5) - (10) for each stage of cell-cycle, with additional terms to account for movement between these stages and the absence or presence of treatment. The resulting model is a system of 30 equations with over 125 individual terms (see Figs. 5 and 6). Although by no means a complex phenomenological description by biological standards, its mathematical expression is incommensurately complex, tedious and fragile. Although they have a long history [12], it is rare to find such elaborate stage-structured compartmentalisation in theoretical biology, not least because of the burden of communication. We intend to automate, and extend, this process.

3.2 Using statecharts to factor model complexity

Figure 3 provides a statechart description of our motivating example, including cell-cycle. Notice how the diagram complexity is closer to that of the biological description. In contrast, the FSM represented by this statechart is shown in Fig. 4 and is of proportional complexity to the

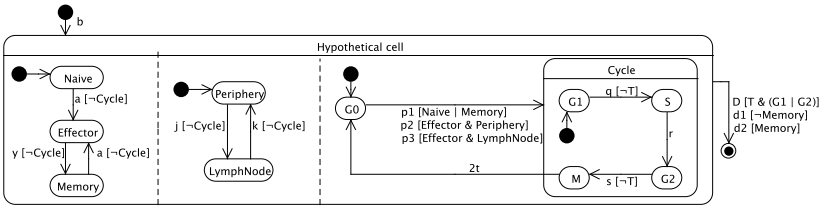


Fig. 3. The statechart representing the illustrative model. Intuitively, this diagram is of comparable complexity to the phenomenological description and requires minimal formal prerequisites to be understood. See text for a discussion of diagram elements.

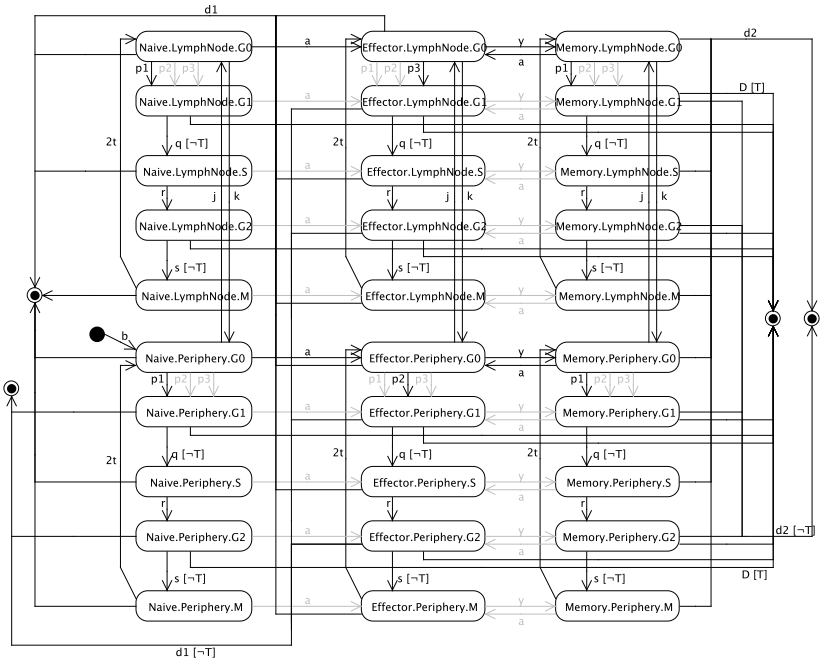


Fig. 4. The underlying directed graph behind Fig. 3. The greyed-out transitions can be statically evaluated as logically invalid and removed prior to simulation. For clarity, we omit invalid transitions into and out of the lymph node. Section 4 describes formally the automatic reduction of statecharts such as Fig. 3 to directed graphs, which is an intermediate step between model formulation and numerical simulation and analysis.

$$\frac{d}{dt}[NP0] = k[NL0] + b + 2t[NPM] - (d_1 + j + p_1 + a([NP0], [MP0], L))[NP0]$$
$$\frac{d}{dt}[NP1] = p_1[NP0] - ((1-T)(d_1 + q) + TD)[NP1]$$
$$\frac{d}{dt}[NPS] = (1-T)q[NP1] - (d_1 + r)[NPS]$$
$$\frac{d}{dt}[NP2] = r[NPS] - ((1-T)(d_1 + s) + TD)[NP2]$$
$$\frac{d}{dt}[NPM] = (1-T)s[NP2] - (d_1 + t)[NPM]$$
$$\frac{d}{dt}[NL0] = j[NP0] + 2t[NLM] - (d_1 + k + p_1 + a([NL0], [ML0], L'))[NL0]$$
$$\frac{d}{dt}[NL1] = p_1[NL0] - ((1-T)(d_1 + q) + TD)[NL1]$$
$$\frac{d}{dt}[NLS] = (1-T)q[NL1] - (d_1 + r)[NLS]$$
$$\frac{d}{dt}[NL2] = r[NLS] - ((1-T)(d_1 + s) + TD)[NL2]$$
$$\frac{d}{dt}[NLM] = (1-T)s[NL2] - (d_1 + t)[NLM]$$
$$\frac{d}{dt}[EP0] = k[EL0] + a([NP0], [MP0], L) + a([MP0], [NP0], L) + 2t[EPM] - (d_1 + j + p_2 + y)[EPO]$$
$$\frac{d}{dt}[EP1] = p_2[EP0] - ((1-T)(d_1 + q) + TD)[EP1]$$
$$\frac{d}{dt}[EPS] = (1-T)q[EP1] - (d_1 + r)[EPS]$$
$$\frac{d}{dt}[EP2] = r[EPS] - ((1-T)(d_1 + s) + TD)[EP2]$$
$$\frac{d}{dt}[EPM] = (1-T)s[EP2] - (d_1 + t)[EPM]$$
$$\frac{d}{dt}[EL0] = j[EP0] + a([NL0], [ML0], L') + a([ML0], [NL0], L') + 2t[ELM] - (d_1 + k + p_3 + y)[ELO]$$
$$\frac{d}{dt}[EL1] = p_3[ELO] - ((1-T)(d_1 + q) + TD)[EL1]$$
$$\frac{d}{dt}[ELS] = (1-T)q[EL1] - (d_1 + r)[ELS]$$
$$\frac{d}{dt}[EL2] = r[ELS] - ((1-T)(d_1 + s) + TD)[EL2]$$
$$\frac{d}{dt}[ELM] = (1-T)s[EL2] - (d_1 + t)[ELM]$$

Fig. 5. Part one of the system of equations for the illustrative example with cell-cycle and treatment. Note that this system of equations corresponds to the directed graph in Fig. 4 in much the same manner as explained in Fig. 1(a). The parameter $T \in \{0, 1\}$ indicates the presence or absence of treatment and is used, like an indicator variable, to implement conditional logic. The three character variable names represent {Naive, Effector, Memory} $\times$ {Periphery, LymphNode} $\times$ {G0, G1, S, G2, M}. We use $[\cdot]$ notation for variables to aid visual clarity, not necessarily to represent concentrations.

$$\frac{d}{dt}[MP0] = k[ML0] + 2t[MPM] + y[EP0] - (d_2 + j + p_1 + a([MP0], [NP0], L))[MP0]$$
$$\frac{d}{dt}[MP1] = p_1[MP0] - ((1-T)(d_2 + q) + TD)[MP1]$$
$$\frac{d}{dt}[MPS] = (1-T)q[MP1] - (d_2 + r)[MPS]$$
$$\frac{d}{dt}[MP2] = r[MPS] - ((1-T)(d_2 + s) + TD)[MP2]$$
$$\frac{d}{dt}[MPM] = (1-T)s[MP2] - (d_2 + t)[MPM]$$
$$\frac{d}{dt}[ML0] = j[MP0] + 2t[MLM] + y[EL0] - (d_2 + k + p_1 + a([ML0], [NL0], L'))[ML0]$$
$$\frac{d}{dt}[ML1] = p_1[ML0] - ((1-T)(d_2 + q) + TD)[ML1]$$
$$\frac{d}{dt}[MLS] = (1-T)q[ML1] - (d_2 + r)[MLS]$$
$$\frac{d}{dt}[ML2] = r[MLS] - ((1-T)(d_2 + s) + TD)[ML2]$$
$$\frac{d}{dt}[MLM] = (1-T)s[ML2] - (d_2 + t)[MLM]$$

Fig. 6. Part two of the system of equations for the illustrative example with cell-cycle and treatment. Continued from Figure 5.

mathematical description in Figs. 5 and 6. Recall, even though this complex model may be what is ultimately *executed*, it is the statechart that is *communicated* and *developed* amongst collaborators.

Certainly, one could use the statechart in Fig. 3 as the basis of a discrete ABM model. But this might be a rather poor and costly choice. It is rare to have sufficient data to properly assess the validity of fine-grained agent behaviours such as movement and interactions. Depending on the size of our hypothetical cell population, simulation and analysis may also be a glacial process. For the remainder, we commit to interpreting the FSM directed graph as an ODE model[4], much like that in Fig. 1(a). What we gain, over Fig. 1(a), is the ability to finesse the homogeneity assumption by introducing fine-grained structuring of the population – approaching ABM heterogeneity without incurring the cognitive costs of maintaining and communicating the accompanying mathematical structure.

4 Reducing statecharts to directed graphs

Although it is well established that statecharts *can be* represented as FSM, the statechart literature has developed several variants of its own

[4] Although we emphasise ODE models, the following techniques are also a valid alternative formalisation of ABM, where agents are encoded as FSM.

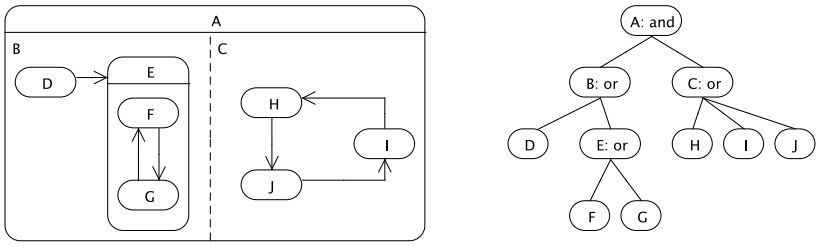


Fig. 7. The structure of a statechart's states can be represented as an AND-OR tree. Nodes in the tree correspond to different state types: simple (leafs), hierarchical (OR branches) and orthogonal (AND branches).

"formal semantics" (e.g. [6, 21]) which do not make use of this connection. Given that the FSM directed graph structure is central to our interpretation as ODE, we now provide what is, to our knowledge, a novel formal description of such a reduction. This not only clarifies the interpretation of diagram elements for the reader, but is sufficient for reproducible, automatic generation of numerical simulations.

4.1 Basic definitions

Let $\mathcal{S}$ and $\mathcal{T}$ represent the sets of all states and transitions in a statechart. The following definitions make use of the fact that the structure of a statechart's states (but not transitions) may be represented as an AND-OR tree [8], as illustrated in Fig. 7. It is this hierarchical structure that is "reduced".

Definition 4.1 (State). *Every state s is represented by the tuple $\{p, v, L\}$, where $p \in \mathcal{S}$ is the parent state in the AND-OR tree and L is an arbitrary label or name. The value v indicates state activity: for FSM, its domain is simply $\{0, 1\}$; for ODE its domain is the real numbers, quantifying the homogeneous sub-population in compartment s.*

Definition 4.2 (OR-state). *An OR state is a compound state that aggregates states hierarchically (see Fig. 2(a)). It is an internal branch node in the AND-OR tree and is represented by the tuple $\{p, a, L, S\}$ where p and L are defined as above and $v = \sum s_i.v$ for each child sub-state $s_i \in S \subset \mathcal{S}$. For one $s_i \in S$ the function* **start** $(s_i) = true$ *indicating the default initial sub-state on entry, which is depicted graphically by the target of the black dot in Fig. 2(a).*

Definition 4.3 (AND-state). *An AND state is a compound state that aggregates orthogonally (see Fig. 2(b)). It too is an internal branch node in the AND-OR tree, however, only OR states are valid child nodes in the tree. Otherwise, it is represented equivalently to an OR state.*

Although they play no part in the AND-OR tree structure, transitions can be seen as additional edges between nodes in the tree.

Definition 4.4 (Transition). *A transition is represented by the tuple* $\{s, t, L, f, g\}$ *where* $s, t \in \mathcal{S}$ *represent the source and target states of the transition, respectively.* L *is an arbitrary label. The function* f *takes different form depending on formalism employed. For FSM,* $f \to \{0, 1\}$ *evaluates whether the currently processed event's name matches this transition's label – the necessary condition for a FSM in state* s *to transition to state* t*. For ODE,* $f(\theta, S) \to \mathbb{R}$ *quantifies the mass transitioning from compartment* s *to compartment* t*. In this case,* f *may be a function of additional states* $S \subset \mathcal{S}$ *and arbitrary parameters* θ*. The function* g *represents a "transition guard", defined below.*

An important expression of "almost independence" is when a particular transition may depend on the specific state of a parallel part of the system in order to be valid. This dependency is realised with *transition guards*.

Definition 4.5 (Transition guard). *A transition guard is a logical expression or indicator function* $g \to \{0, 1\}$ *that constrains the validity of a transition. Diagrammatically, they are traditionally presented beside the transition label in square brackets (see Fig. 3). The expression* g *may refer to parallel states by label* $s.L$*, evaluating presence or inequalities in* $s.v$*.*

Such guards can be very fine-grained when evaluated in terms of an individual in a FSM or ABM. For an ODE model, their applicability is more limited but still meets several important use-cases. For example, in Fig. 3 transition guards add realism (or make assumptions explicit) by restricting cells who have entered the cell-cycle from moving or changing functional state. In cases such as this, transition guards can be statically resolved, prior to simulation, because their logical validity depends on where in the directed graph they are evaluated (e.g. greyed-out transitions in Fig. 4). Other guards, that reference parameters or make use of inequalities, may need to be evaluated during simulation.

4.2 State reduction

The process of reduction transforms the AND-OR tree into a flat directed graph with only simple states (nodes) and transitions (edges).

This is always well-defined, because the statechart contains a complete description of its architecture. Starting from the root of the AND-OR tree, the following procedures are carried out recursively in depth-first order for each state s

Definition 4.6 (State reduction). reduce $(s) = s$

Definition 4.7 (OR-state reduction). reduce $(s) = \bigcup_{s_i \in s.S} \mathbf{reduce}(s_i)$

In the case of orthogonal states, the reduction is a little more involved. First, we need to introduce a state type that is never manipulated by the modeller:

Definition 4.8 (Product state). *A product state is a basic state that represents an element in the Cartesian product of sub-states of each OR-state $s.S_i$ that are child nodes of an AND-state s. The product state is represented by the tuple $\{p, v, L, P\}$ where $p = s$ and v is a scalar just as for the basic state. The set P contains the particular combination of parallel states that this product state represents. The label L is a concatenation of the labels $p.L$ for $p \in P$.*

Definition 4.9 (AND-state reduction). reduce $(s) = \bigcup_i \pi_i$ *where* π_i *is a product state with* $\pi_i.P \in \mathbf{reduce}\,(s.S_i) \times \ldots \times \mathbf{reduce}\,(s.S_n)$

By our convention, the root of the tree (the statechart) is an AND-state, thus all nodes in the directed graph are product states. Letting $\mathcal{Z}$ represent the set of reduced states, then $\mathcal{Z} = \mathbf{reduce}\,(c)$ for some statechart c. Having produced the appropriate graph nodes, we now construct the transition topology that preserves the statechart's semantics.

4.3 Transition reduction

Along with $\mathcal{Z}$, let $\mathcal{Y} = \emptyset$ represent the set of reduced transitions. Then, assuming no additional constraints on transitions, the basic reduction process is as follows.

Definition 4.10 (Transition reduction). *For each transition $t \in \mathcal{T}$ and each pair of states a, b in $\mathcal{Z} \times \mathcal{Z}$, $\mathcal{Y} = \mathcal{Y} \cup \mathbf{rewire}\,(t, a, b)$ if* **connected** (t, a, b)

where **rewire** (t, a, b) produces a copy of t with a and b as end-points and **connected** (t, a, b) is defined as

Definition 4.11 (Product state connectivity). *Given a transition* t *and two product states* a *and* b, $\mathbf{connected}\,(t, a, b) \rightarrow \{true, false\}$ *is true if and only if* $a.P - b.P = t.s$ *and* $b.P - a.P = t.t$. *That is, the product states have the same elements except for one and the anomalous elements are the transitions end-points. Connectivity is not symmetric.*

However, transitions with compound states as end-points require special handling because these states (such as *Cycle* in Fig. 3) are not explicitly retained during the reduction. The statechart semantics of such transitions is intuitive enough (see e.g. Fig. 2(a)) but preserving these semantics across arbitrary statecharts is notoriously difficult and many formal treatments impose additional restrictions on what makes a valid statechart (see e.g. [17, 13]). The following definition is, at least, sufficient when parallelism is restricted to a top-level organisational construct. For biological models, this is the most plausible use-case.

Definition 4.12 (Transition reduction with compound states).

For each $t \in \mathcal{T}$

1. *Let* $\mathcal{S}_t = \emptyset$ *and* $\mathcal{T}_t = \emptyset$ *represent candidate source and target states*
2. *Unless* $\mathbf{isCompound}\,(t.s)$ *then* $\mathcal{S}_t = \mathcal{Z}$.
   Else $\mathcal{S}_t = \{z : z \in \mathcal{Z}$ *and* $\mathbf{isChild}\,(t.s, z)\}$.
3. *Unless* $\mathbf{isCompound}\,(t.t)$ *then* $\mathcal{T}_t = \mathcal{Z}$. *Else*
   (a) *If* $\mathbf{isParallel}\,(t.t)$ *then*
   $\mathcal{T}_t = \{z : z \in \mathcal{Z}$ *and* $\mathbf{isChild}\,(t.t, z)$ *and* $\mathbf{start}\,(z)\}$.
   (b) *If* $\mathbf{isHierarchical}\,(t.t)$ *then*
   $\mathcal{T}_t = \{z : z \in \mathcal{Z}$ *and* $\exists s_i \in z.P$ *where* $\mathbf{isChild}\,(t.t, s_i)$ *and* $\mathbf{start}\,(s_i)\}$.
4. *For each pair of states* a, b *in* $\mathcal{S}_t \times \mathcal{T}_t$,
   $\mathcal{Y} = \mathcal{Y} \cup \mathbf{rewire}\,(t, a, b)$ *if* $(\mathbf{isCompound}\,(t.s)$ *or* $a.P - b.P = t.s)$ *and* $(\mathbf{isCompound}\,(t.t)$ *or* $b.P - a.P = t.t)$.

where predicates $\mathbf{isCompound}\,(\cdot)$, $\mathbf{isHierarchical}\,(\cdot)$ *and* $\mathbf{isParallel}\,(\cdot)$ *are self-explanatory. For regular states* x *and* y, $\mathbf{isChild}\,(x, y)$ *searches up the AND-OR tree, from* y, *for a parental state* x. *For product states,* $z \in \mathcal{Z}$, *which are not part of the statechart or AND-OR tree,* $\mathbf{start}\,(z) = \bigwedge_i \mathbf{start}\,(z.P_i)$ *and* $\mathbf{isChild}\,(\cdot, z) = \bigvee_i \mathbf{isChild}\,(\cdot, z.P_i)$. *Notice that Def. 4.12 reduces to Definition 4.10 for transitions between two simple states.*

Unfortunately, this transition reduction is not quite sufficient to properly express important statechart semantics. Another common dependency between "almost independent" states is orthogonal transitions that respond to the same event, causing synchronous state changes in

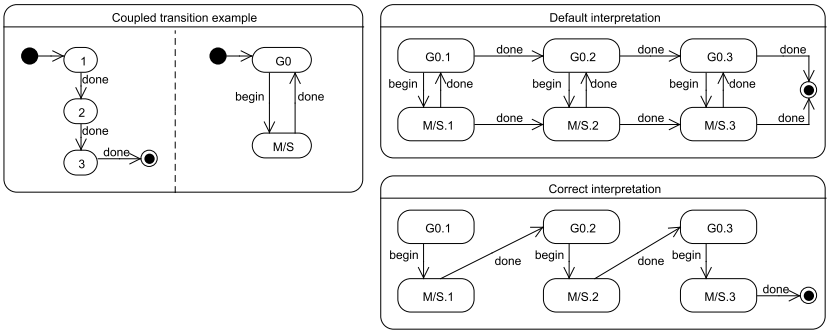


Fig. 8. An illustration of the semantics of a coupled transition (*done*) in parallel charts, representing a simple cell division model (left). As daughter cells transition from M/S-phase they should *also* move through the numbered generational compartments. The default interpretation of Def. 4.12 (top-right) does not respect this, motivating Def. 4.13.

orthogonal charts. Figure 8 illustrates the problem with a parallel statechart representing a simple cell division and 3-generation stage structuring. Clearly, these are not independent: when (a population of) daughter cells transition from the S/M-phase back to state $G0$ they should *also* move through the generational stages. The right half of Fig. 8 illustrates the differences in the directed graph generated from the reduction above and the one intended by the statechart semantics. The result is still a single transition, but the topology of the FSM graph is now quite different.

Fortunately, if transitions with the same label occur in parallel charts, then they will always occur multiple times as outgoing transitions from any product state that represents the combination of their individual sources. Such a situation is *never* logically valid for ODE (or deterministic FSM), making for an non-determinism we can exploit to identify and correct erroneous reductions.

Definition 4.13 (Coupled transition correction).

To identify and correct erroneous reductions in Def. 4.12

1. *Let* $\mathcal{R} = \mathcal{D} = \mathcal{L} = \emptyset$
2. *For each* $t_1, t_2 \in \mathcal{Y} \times \mathcal{Y}$*, if* $t_1.s = t_2.s$ *and* $t_1.L = t_2.L$ *then* $\mathcal{R} = \mathcal{R} \cup t_1.s$*,* $\mathcal{D} = \mathcal{D} \cup t_1$ *and* $\mathcal{L} = \mathcal{L} \cup t_1.L$*.*
3. $\mathcal{Y} = \mathcal{Y} - \{t : t \in \mathcal{Y} \text{ and } t.L \in \mathcal{L}\}$*.*
4. *For each* $s \in \mathcal{R}$

(a) *Let* $P = \emptyset$
(b) *For each* $t \in \{t : t \in \mathcal{D} \text{ and } t.s = s\}$ *do*
$P = P \cup (t.t.P - s.P)$
(c) *For any* $t \in \{t : t \in \mathcal{D} \text{ and } t.s = s\}$ $\mathcal{Y} = \mathcal{Y} \cup \mathbf{rewire}(t, s, z)$ *for each* $z \in \{z : z \in \mathcal{Z}$ *and* $P \subset z.P\}$.

The intuition behind the last step is that we are able to sufficiently reconstruct $z.P$ for the correct target product state(s) z, even if transitions are only coupled in a subset of parallel states.

4.4 Some minor details

The reduction is now complete. For brevity, we omit the details of a final check for (*i*) transitions that exit terminal states, (*ii*) transitions with guards that are always false, and (*iii*) states with no incoming transitions. Lastly, because none of the original states in the statechart will make it into the reduced model, we have found it convenient to introduce *implicit states* that serve no role in the reduced model or its simulation, except to provide aggregated measures across the appropriate subset of $\mathcal{Z}$ that represents an original state in the statechart.

5 In-silico analysis and exploration after reduction

Given the reductions defined in Sect. 4, the resulting directed graph is readily analysed and simulated, either in-place or by translation to an existing format. Although we have no intention of offering the illustrative example of Sect. 3 as a proper model, for completeness we briefly demonstrate the type of analysis that is possible, which allows us to address a pertinent question raised by a reviewer: *at what point does the ability to describe complex models with relative ease become a hindrance to understanding the system in question?*

From a practical perspective, the analysis of statechart-derived models can also benefit from the combinatorial factorisation. Because we simulate all low-level state combinations, it is possible to reconstruct any subset of the population by aggregating these combinations – e.g. in Fig. 9, abusing notation slightly $[Memory]_t = \sum_i \sum_j [Memory.S_i.S_j]_t$. Thus, we are able to examine (or ignore) how heterogeneity in the subpopulations influence the homogeneous populations in the phenomenological description (e.g. Fig. 10). Similarly, we are able to construct projected views of the model's dynamics that omit or cut across certain subsystems, e.g. highlighting differentiation in particular locations (Fig. 10). This ability to construct views of the factorised model is crucial:

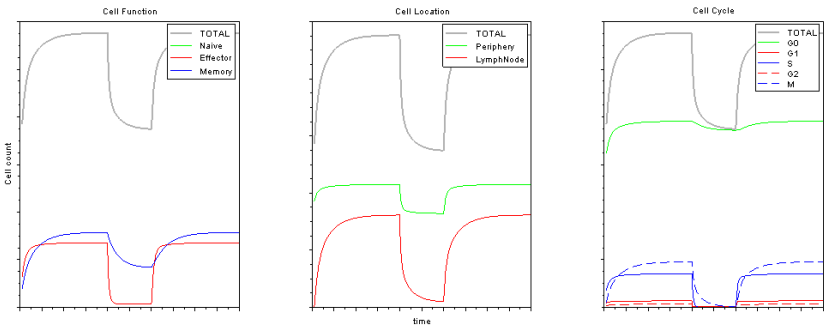


Fig. 9. The simulated dynamics of the states represented in the statechart Fig. 3. Each graph corresponds to a parallel subsystem, each curve to a simple state in that subsystem. The model is run until steady-state then treatment and recovery are simulated as described in Sect. 3. Biological interpretation should be avoided and axes markings have been removed to emphasise that model parameters and axes scales are arbitrary.

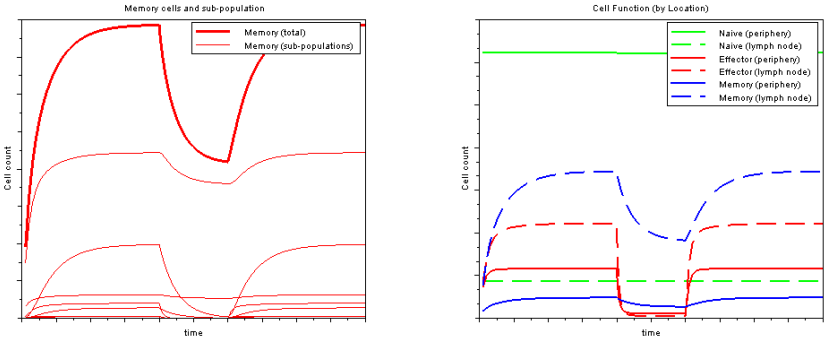


Fig. 10. Constructing "views" of the system dynamics from kinetic data of the low-level state combinations. **Left:** how the heterogeneous dynamics of memory cell sub-populations contribute to the curve of the *Memory* compartment in Fig. 9. **Right:** how functional differentiation differs by location, without regard for cell-cycle. Such views allow us to better understand the coarse dynamics in Fig. 9 and compare to, or predict, experiments that cannot simultaneously observe the entire system.

although parallel sub-systems may be "almost independent" in principle, in practice, it is likely that experimental data is not. This is particularly true in cellular biology, where observing different aspects of a system can involve entirely different experiments or treatments that interfere with observations. One can easily envisage a situation where, for a diagram such as Fig. 3, the formulation of parallel sub-systems and the supporting experimental data has been produced by remote labs, each with their own motivation for studying the system. From the "systemic" perspective presented here, it is apparent that independent models of parallel sub-systems, like the apocryphal blind men and the elephant, risk misrepresenting the quantitative influence of dependencies through a mixture of erroneously inferred rates or mechanisms and appeals to experimental variance. Although we make no claim to solving this problem, the statechart formulation provides a clear path towards such systemic models – possibly bootstrapping from independent studies – by constraining projections of the model that admit empirical support. Likewise, novel projections may provide testable predictions.

From a theoretical perspective, observe that although we emphasised in Sect. 3 that the structural complexity of a systemic model grows multiplicatively, the parametric complexity only grows additively. It is implicit in the statechart formulation that transitions are *replicated* and parameters *reused*. There are two separate numerical issues here: the number of parameters that need to be searched (the domain) and the "shape" of the error function to be optimised (the range). A combinatorial increase in states may not substantially affect the former, however, we expect identifiability issues in the latter [2]. Our research is simply not mature enough to offer any real insights into such problems at this time, but we are inclined to concur with [23] that ill-posedness is a simple reality of biological modelling. The non-uniqueness of a model's solution provides useful information that may elicit prior knowledge or direct empirical research.

Lastly, a compelling feature of this graphical formalism, not discussed in the preceding material, is its malleability and extensibility. States, transitions and parameters are all first-class objects that can be easily rewired and reparameterised even, in principle at least, by experimentalists. State and transition classes can be extended to represent e.g. arbitrary non-linear functional forms and common modelling patterns such as temporally-constrained states. This allows theorists and experimentalists to explore structural and functional forms of a model without maintaining the underlying numerical realisation. Although existing graphical modelling languages claim similar benefits, to our knowledge

none are able to generate the substantial bulk of a model automatically from a concise human-friendly description.

6 Discussion and future work

We submit that with some minor enrichment of the statechart diagram language, to better align it with scientific investigation and reporting, statecharts could serve as an effective medium of communication and development for both theorists and experimentalists. We have only concerned ourselves with a formalisable subset of statecharts proper – avoiding features unnecessary for our needs and, arguably, for scientific models. Although we have presented the formal constructs in the context of an hypothetical example, much of this example is based on our experience with difficulties in modelling immunological data that is inherently parallel and multi-level using traditional techniques. This work is still in progress, though we have some idea where future efforts may focus.

Diagram enrichment Our non-classical use of statecharts leaves us wanting in some respects. For example, the informal referencing of parameters, variables, their units and the functional form of transitions in Fig 3. Largely, these issues are not ODE-specific and appear to be solvable with "syntactic sugar" rather than reinventing the statechart formalism under a different moniker.

Computational complexity Our focus has been on managing descriptive complexity, but computational complexity is an ever-present issue too. It is an interesting open question as to when highly compartmentalised ODE models would become less efficient than a pure ABM effort[5]. Certainly, although we can insulate the modeller and the experimentalist from the combinatorial growth of states, we cannot yet insulate the machine. For scientific simulation, this is a much less pressing issue than depicted in the statecharts literature, where systems are often embedded with realtime constraints, but exponential growth is still a concern in the long term. One interesting possibility is that the techniques used by statechart researchers (e.g. see [22]) may be adapted as hybridised numerical integration algorithms.

[5] A rough guess would be when the number of state combinations exceeds the number of agents, although such a situation seems largely artificial as subsequent statistical analysis would need to smooth over this fine level of state granularity.

Coupled states A fundamental aspect of complex systems is the interactions of non-trivial entities. For ODE modelling, the coupling of molecular species into complexes is well established using techniques from chemical kinetics. However, molecules are simple stateless objects. As a contrasting example, different immune cells undergo surface-binding while still progressing through the "almost independent" processes of metabolism, regulation and so on. Classically, models of cellular interactions are reduced to the chemical kinetics of receptor-ligand binding – treating the cell as an implicit object. To fully express interactions between complex entities, what is required is the expression that some states are coupled between different systems during interactions. To the best of our knowledge, the coupling of state machines is a surprising omission in the statechart literature (although, see e.g. [10]). We have largely solved this problem for pairwise coupling (not shown), but not in general for n-wise coupling.

Methodological unification Statecharts can represent high-level semantics suitable for pedagogical descriptions as well as low-level quantitative information suitable for individual-based ABM (traditionally) and population-based ODE (shown here). A unified formalism would better allow these different levels of description to be directly compared, highlighting how each modelling effort realises the phenomenological description and the quantitative and structural assumptions it makes in doing so. This would require some novel enhancements to the statechart language, e.g. abstracting interactions that may be based on mass-action assumptions or localised cellular-automata neighbourhoods, depending on the simulation method. This is largely a problem of transition implementation, although there are several subtle issues that would need to be resolved.

7 Conclusion

The ability to model sub-system parallelism and multi-level hierarchy is a powerful feature of Harel statecharts – a feature that is sometimes lost in the literature under computer science and software engineering nomenclature. We have shown how a formalised subset of Harel statecharts can be applied to scaling mathematical descriptions of systemic phenomena. In addition to finessing the associated increase in underlying model complexity for the modeller, this approach provides a graphical communication medium that, in our experience, is readily accepted by non-technical collaborators and can be productively discussed, questioned and reformulated without excessive concern for underlying technical details.

We submit that this technique complements existing methods in Systems Biology, allowing a systems approach to be taken at a coarser granularity than biochemical reactions. Such progress is necessary in many domains, such as immunology, where the primary focus is on intra- *and* inter-cellular interactions. Of course, this argument generalises to the scientific study of any complex system where "agents" are compound, stateful and inter-dependent.

References

[1] Marco Ajelli, Bruno Gonçalves, Duygu Balcan, Vittoria Colizza, Hao Hu, José J Ramasco, Stefano Merler, and Alessandro Vespignani. Comparing large-scale computational approaches to epidemic modeling: agent-based versus structured metapopulation models. *BMC infectious diseases*, 10:190, January 2010.

[2] R. Bellman and K. J. Astrom. On structural identifiability. *Mathematical Biosciences*, 7:329–339, 1970.

[3] Sol Efroni, David Harel, and Irun R Cohen. Emergent dynamics of thymocyte development and lineage determination. *PLoS computational biology*, 3(1):e13, January 2007.

[4] Daniel T. Gillespie and Linda R. Petzold. *Numerical simulation for biochemical kinetics*. MIT Press, 2006.

[5] D Harel. Concurrency in Biological Modeling: Behavior, Execution and Visualization. *Electronic Notes in Theoretical Computer Science*, 194(3):119–131, 2008.

[6] D. Harel and A. Naamad. The STATEMATE semantics of statecharts. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 5(4):293–333, 1996.

[7] David Harel. Statecharts: a visual formalism for complex systems. *Science of Computer Programming*, (8):231–274, 1987.

[8] David Harel. Statecharts in the Making : A Personal Account. In *HOPL III Proceedings of the third ACM SIGPLAN conference on History of programming languages*, 2007.

[9] T. A. Henzinger. The Theory of Hybrid Automata. In *Proceedings of the 11th Annual Symposium on Logic in Computer Science (LICS)*, pages 278–292. IEEE Computer Society Press, 1996.

[10] Dominikus Herzberg and Andre Marburger. An extension of state machine modeling: the concept of coupled state machines. In *OMER-2 Workshop on object-oriented modeling of embedded RT-systems*, 2001.

[11] Hiroaki Kitano, Akira Funahashi, Yukiko Matsuoka, and Kanae Oda. Using process diagrams for the graphical representation of biological networks. *Nature Biotechnology*, 23(8):961–966, 2005.

[12] L. P. Lefkovitch. The study of population growth in organisms grouped by stages. *Biometrics*, 21:1–18, 1965.

[13] Florence Maraninchi and Yann Remond. Argos: an automaton-based synchronous language. *Computer Languages*, 27:61–92, 2001.

[14] John H. Miller and Scott Page. *Complex Adaptive Systems*. Princeton University Press, 2007.

[15] Sarah P. Otto and Troy Day. *A biologist's guide to mathematical modeling in ecology and evolution*. Princeton University Press, 2007.

[16] Yong-Jun Shin and Mehrdad Nourani. Statecharts for gene network modeling. *PLoS ONE*, 5(2), 2010.

[17] A. J. H. Simons. On the Compositional Properties of UML Statechart Diagrams. In *Rigorous Object-Oriented Methods*, 2000.

[18] Michail Stamatakis, Kyriacos Zygourakis, and Monte Carlo. A mathematical and computational approach for integrating the major sources of cell population heterogeneity. *Journal of Theoretical Biology*, 266(1):41–61, 2010.

[19] Naamah Swerdlin, I.R. Cohen, and David Harel. The lymph node B cell immune response: Dynamic analysis in-silico. *Proceedings of the IEEE*, 96(8):1421–1443, 2008.

[20] Veronique Thomas-Vaslin, Hester Korthals Altes, Rob J. de Boer, and David Klatzmann. Comprehensive Assessment and Mathematical Modeling of T Cell Dynamics and Homeostasis. *The Journal of Immunology*, (180):2240–2250, 2008.

[21] Michael von der Beeck. A Comparison of Statecharts Variants. In *Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 128–148, 1994.

[22] Andrzej Wasowski. Flattening Statecharts without Explosions. *ACM Sigplan Notices*, 39(7), 2004.

[23] Sven Zenker, Jonathan Rubin, and Gilles Clermont. From inverse problems in mathematical physiology to quantitative differential diagnoses. *PLoS Computational Biology*, 3(11), 2007.

Plotting a catchy tune: tracing sound meme evolution through visualization

A. Guest[1], J. Bown[2], A. Sapeluk[1], A. Winfield[3], and M. Shovman[2]

[1] School of Computing and Engineering Systems, University of Abertay Dundee, UK
[2] Institute of Arts, Media and Computer Games, University of Abertay Dundee, UK
[3] Bristol Robotics Laboratory, University of the West of England, Bristol, UK
m.shovman@abertay.ac.uk

Abstract. Complex systems comprise many relative simple agents interacting in space over time. The interactions among the agents can give rise to emergent behaviours that are not deducible from the individual agents alone. To understand the dynamics of complex systems is challenging and visualization is a common tool used to aid understanding. We present a case study of a complex system focused on communication among agents. We have developed visualizations for intra-individual state, inter-individual communications and community-scale diversity in communication. We report on the purpose, approach and insights gained from each visualization in turn. We also present a critical appraisal undertaken by a party with expertise in visual analytics and external to the project, and report candidly on the limitations of our visualizations. This includes us creating limited, misleading and confusing interpretations as we translate from the visual representation back to the data domain. We also propose some improvements to these specific visualizations and offer some general guidelines for effective visualizations.

1 Introduction

Complex systems are known to comprise many individual agents interacting in an environment, and be driven by processes that operate at

multiple scales in space over time [12]. In many systems these individuals are all different, and this variation is important for system dynamics [4]. Moreover, global, or at least non-local, emergent behaviour may arises from the local interactions among individual agents [6]. Processes at multiple scales, emergent behaviour, individual variation, combined with the difficulties associated with system measurement generally [7], means that characterizing system dynamics is challenging. Modelling can aid understand as to how behaviours at the individual component scale give rise to emergent patterns at larger scales, and how those larger-scale phenomena impact on the behaviours of smaller-scale individuals [4]. However, interpreting model dynamics at multiple scales is likewise challenging, and as systems get larger in space and time this becomes increasingly difficult.

Here, and detailed in Section 2, we chose as a candidate system the artificial culture laboratory [20] designed to explore cultural evolution through changes to memes, i.e. units of cultural transmission [5]. This artificial culture laboratory implements a basic architecture that affords (re)production of memes, variation in meme production and a range of meme selection strategies. Through these fundamental processes, we are able to identify conditions that promote and inhibit both meme diversity and reproductive fidelity. The agents in this system are physical, mobile robots that exchange memes through inter-robot interactions. As described in Section 2, these real robots are heterogeneous and communication is error-prone and highly spatially contingent. Our interest in this system pertains to identifying conditions that promote effective meme diversity within and propagation across a community of interacting robots, and observing any emergent behaviour linking observed community-scale.

To explore this, we are interested in phenomena at three different scales: individual robot memory, i.e. the set of memes that a robot has stored; inter-robot communication, i.e. occurrences of meme exchange; and community memory, i.e. the set of memes in the collective robot population. Because of the large number of meme exchanges, errors in meme exchange and variation in memory among individuals interpreting the system dynamics is challenging. Indeed, this is a generalisable challenge for any system of heterogeneous, interacting agents: we may be interested in internal state, interactions between individuals and the impact of those interactions on community state.

Visual representations may help this interpretation; human visual perception routinely copes with large amounts of input data, effortlessly parsing complex and confusing sensory stimuli into coherent and meaningful perceptual objects. Specifically, pattern-recognition abilities of hu-

man cognition make visualisation an effective method for understanding complex models [18]. Visual exploratory data analysis is concerned with detecting and describing pattern and relations in data [1], such as outliers, trends and clusters [3]. The idea of visualising abstract data for analysis and exploration is not new [11], but a focused research in visualisation design and usage, as a separate discipline of Visual Analytics, has only recently started to be acknowledged as a separate area of scientific endeavour in its own right, helping analysts to detect the expected and discover the unexpected [16].

Visual Analytics seeks to ensure that visualisations are efficient and fit for purpose, since poor visualisations may offer no insight or worse still mislead (e.g. [8]. This is made possible by advances in understanding processes of human perception and cognition [18]. Visual Analytics supports the design of tools that presents data in a way that is optimised for human perception, and provides objective benchmarking of tool performance in terms of efficacy in conveying relevant information.

In the most general terms, data visualization is a process of creating images, diagrams or animations based on abstract data to convey information efficiently [17]. A mathematical formalism exists [9] that represents visualizations as functions mapping from data space into visualization space. Thus, for example, in a yearly precipitation chart, date and precipitation levels are mapped onto horizontal and vertical axis of the chart respectively. The same mapping function maps the tasks the analyst must undertake in the data domain, e.g., to find the rainiest month of the year, to visual tasks, e.g., to find a month-sized region of the chart with the largest area under it. Likewise, the domain-specific insights gained from visual analysis are mapped from visual tasks, e.g., a single sharp peak in a generally low flat region, back into the data domain as an interpretation, e.g., either a sudden shower in the middle of a dry season – or an error in data collection.

Analysing of the effectiveness of visualizations in terms of the relation between visual tasks and domain data insights is a promising novel approach [14]. It is favourably comparable with previous analysis methodologies such as user studies [18] or the analysis of constituent visual elements (e.g. [17, 18, 3]. The main benefits are that it draws upon the wealth of research in visual perception [15, 18] and is easily generalisable across subject domains,. For instance, the majority of visual analytics tasks, when expressed in visual terms, can be parsimoniously divided into only three groups: outlier detection, clustering (group detection) or trend detection [15].

Here, we consider the use of visualisation to understand the relation between scales in this particular complex system, especially how

individual interactions drive large-scale dynamics. We consider how to describe visually the memory of individuals, especially to identify novel meme structures identified through errors, and the phenomena of interest: community-scale meme diversity and meme propagation. These visualisations at different scales may be a useful mechanism for describing scale-linkages, and this is of particular value when systems are scaled up because the system dynamics become increasingly difficult to elicit [13].

2 The Artificial Culture Lab

The artificial culture lab comprises a physical arena with closed boundaries, populated by mobile robots called e-Pucks (www.e-Puck.org), that are wheeled and capable of moving forwards, backwards and turning [10]. They are equipped with a range of sensors that enable detection of obstacles and other robots. Robots can signal to each other with movement and light (through programmable LEDs), and both movement and light may be detected through a simple on-board camera. Robots can also signal to each other through sound, as each has an on-board speaker and microphone.

We have already used this tracking system in a study on movement. In this work [19], memes are self-contained movement features. While a teacher robot, seeded with one or more initial memes, enacts its meme, one or more learner robots observe that meme and store it in memory. When learner becomes teacher, a meme is selected from memory and enacted while other learner robot(s) observe. Importantly, we preclude robot-to-robot telepathy: the learner robot formulates the meme enacted by the teacher through its senses alone.

In our case study, we focus on sound-memes: short tunes generated through on-board speakers and heard through on-board microphones. Crucially, this system is noisy and robot-to-robot communication is so difficult that we had to resort to simulation. Robots were not able to generate a consistent frequency, and need to be directly facing the singing robot under idealized (sound proofed) conditions.

Moreover, there is a strong distant-dependent attenuation, with the strength of signal dropping off sharply to 25% at 10cm and 10% at 15cm, with a linear decay until 70cm. We accommodated this in the simulator via systematic in vitro experiments to characterize that natural variation and then parametrise the simulator based on those experiments as in previous ecological studies [4].

The simulator is a bespoke simulation developed for this research. It is a high-fidelity simulation of the audio aspects of the e-Puck robots,

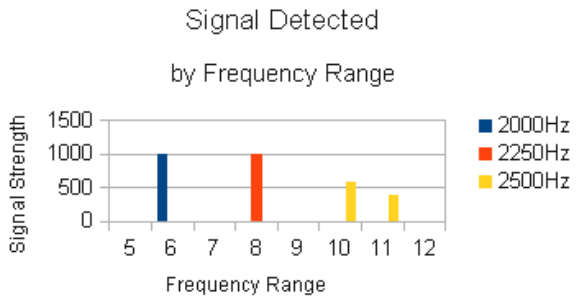


Fig. 1. A sample of how actual frequency maps to frequency ranges. 2000Hz and 2250Hz give clear, strong signals. 2500Hz is interpreted as two distinct, weaker signals in two different frequency ranges.

replicating the sound generation and detection capabilities of the e-Puck robots, specifically; the sound attenuation over distance, the range of sounds that can be generated and detected and the variation in frequency of an e-Puck making a sound of a "constant" frequency. The directionality of the real e-Pucks sound detection is not simulated. In addition the simulator is a low-fidelity simulation of the movement of the e-Pucks.

The e-Pucks cannot distinguish which e-Pucks they are hearing, they rely on the frequencies they hear to distinguish one meme from another. The e-Pucks analyse the sounds they hear and break them down in to different frequency ranges and treat the sounds made within a single range as a single meme. This allows the e-Pucks to distinguish two different memes sung by two different e-Pucks at the same time provided the memes are sung within different frequency ranges. If two e-Pucks sing two different memes at the same time, in the same frequency range, then any e-Puck hearing them will be unable to distinguish the two memes, instead the listener will perceive a single meme consisting of the combination of the two sung memes.

Due to the variation in frequency (discussed above), it is possible that a single e-Puck trying to sing in one frequency range will sometimes sing part of the meme in an adjacent frequency range. When this happens a listener will perceive two distinct memes in two distinct frequency ranges, usually one meme is very close to the meme intended to be sang and the other consists of small snippets of that meme.

Importantly, there is no synchronization between these robots. Additionally, it is not possible (by design) for a robot to distinguish between

different robots singing and consequently, memes may be heard individually or misheard in combination, i.e. memes overlapping since there is no special delimiters on memes. Further, the inherent noise in the system means that some memes may be mis-sung, i.e. generated with errors such that the listened for frequency is not generated. Every robot starts a given simulation with a small set of pre-seeded tunes in memory. Robots move around an arena, listen to tunes sung by other robots and then imitate what has been heard, under different meme selection strategies.

Several visualizations have been developed to analyse robot meme fidelity and propagation at three different scales: individual robot meme memory, community-scale meme diversity over one or more simulations and individual-to-individual meme exchange in a single simulation. These visualizations will be presented and analysed in a format of case study: for every stage of the data analysis: the research questions will be described; the visualizations employed to address these questions outlined; the insights that were obtained using these visualizations, both in visual and in data terms; and in the light of all the above, a critical analysis of the role of the visualization technique will be presented. This critical analysis has been undertaken by someone external to the case study and so is as objective as possible. In the spirit of Visual Analytics motto, Detect the expected and discover the unexpected [16], the answers presented will be of two kinds: expected answers to the questions asked; and unexpected insights generated by the visual analysis.

3 Individual e-Puck Memory

3.1 The Question

What are the memes heard by an individual robot, and how does this vary from the original seed meme?

The aim of the first analysis was to provide an overview of the range of memes generated during a three-robot experiment. All three robots, each seeded with the same initial meme, began communication, listening and copying each other's memes. Importantly, there is no synchronization between these robots. Since memes may be misheard and mis-sung, variety will occur during the simulation, and we are interested in the nature of that variety. Here, we considered a simple meme memory and meme selection strategy. When a robot heard a meme it was added to memory, regardless of whether the meme had been heard before. For selection of a meme, robots picked memes at random [19] from memory, and so memes that have been heard often are more likely to be selected again.

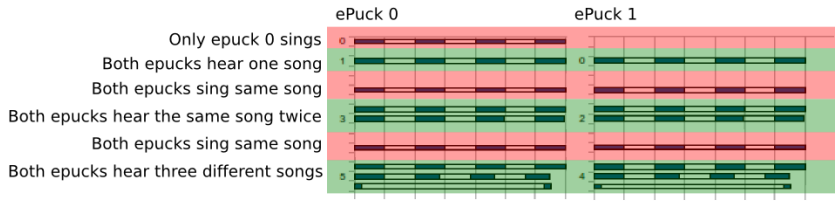


Fig. 2. A sample of a beginning of two-robot simulation run. Every bar represents an individual tune (meme), with darker areas indicating sound pulses and lighter areas indicating pauses.

Figure 2 above shows the beginning of an imitation experiment with two e-Pucks. It alternates between showing what is being sung and being heard. The experiment starts with only one e-Puck (0) singing. Both e-Pucks 0 and 1 hear that meme. Each responds by randomly singing one of the memes it knows. The final heard section of memory shows that one of the e-Pucks must've has mis-sung at the beginning and end of the meme leading to a new meme being heard in a different frequency range.

3.2 The Visualization

All the tunes stored in a robot's memory by the end of a single run were visualized, separately for each robot. Figure 3 (left) shows data from a single robot listening and mimicking. In this visualization, every horizontal line is an individual meme, with dark grey areas indicating sound pulses and light grey areas indicating pauses. The lines are stacked vertically, sorted by number of pulses in a meme (two to five), then by overall tune length, then by the length of the first note. The horizontal axis shows time in milliseconds, indicating the total length of the meme.

3.3 The Interpretation

This visualization allows perceptual grouping of similar adjacent lines and therefore separation of the whole meme set into a small number of clusters of similar memes. The diversity of memes can clearly be seen, together with the similarity of meme structures within clusters.

One insight generated by this visualization was the appearance of groups of similar memes that exhibit a trend of small changes in overall meme length. It is conceivable that the tunes, being misheard, grow progressively longer (or shorter), mutating over time. Note that the visualization used in Figure 3 (left) loses the time component of the experiment. In order to explore the hypothesis that tunes get shorter over

time, the data was re-plotted in the order of being recorded in robot's memory (Figure 4 (right)). This presentation of data immediately shows that tune length varies over time, and that although some shortening does occur (the first two lines are the longest, even if by a short margin), no consistent trend of decrease in tune length is present.

Another insight is the generation of very long memes with long silences in between two short pulses, and also very short memes. Both arise from the asynchrony and lack of delimiter in robot-to-robot transmission. Since there are no delimiters, robots may begin listening to another robot at the final pulse of one meme and, by chance, stop listening at the end of another meme sung by perhaps another robot. This asynchrony can generate novel meme structures.

Figure 4 shows the memory of all three individual robots, and the effect of sound attenuation. The three robots are aligned equidistantly in a row, 5cm apart. Each e-Puck produces tunes at a different frequency, in a different frequency range. The right hand e-Puck's frequency is at the lower end of its frequency range. This means the frequency variation will affect this e-Puck more than the others. When a listener e-Puck processes what it hears from this e-Puck, some of the received signal will fall in the frequency range below the one it was produced in. Normally the amount of mis-heard signal is below the threshold for detection, and is therefore ignored, but it does mean that the signal in the correct range is weaker and therefore is more affected by the attenuation of the signal over distance. The result is that the left hand e-Puck cannot always hear the right hand e-Puck. The left and centre e-Pucks sing strongly in the centre of their frequency ranges so can always be heard by all three e-Pucks.

The effects of this can be seen in Figure 4. The centre e-Puck sings clearly and is easily heard by the others. It hears everything the other e-Pucks sing (including itself) and picks up any mis-sings by the right hand e-Puck. Consequently it heard more memes than any other e-Puck. The right hand e-Puck is not a clear singer, its 'voice' is weaker and more prone to errors caused by frequency variation. Note, this is a clear example of the variation occurring as a consequence of using real robots. However it can hear all the e-Pucks clearly, including itself, since it is close enough to itself to pick up its own weak signal. As the result, it has heard almost as many memes as the centre e-Puck. The left hand e-Puck sings clearly enough to make itself heard by all the others. It can hear itself and the centre e-Puck well. However it sometimes fails to hear the right hand e-Puck, and therefore has heard less memes than the other two e-Pucks.

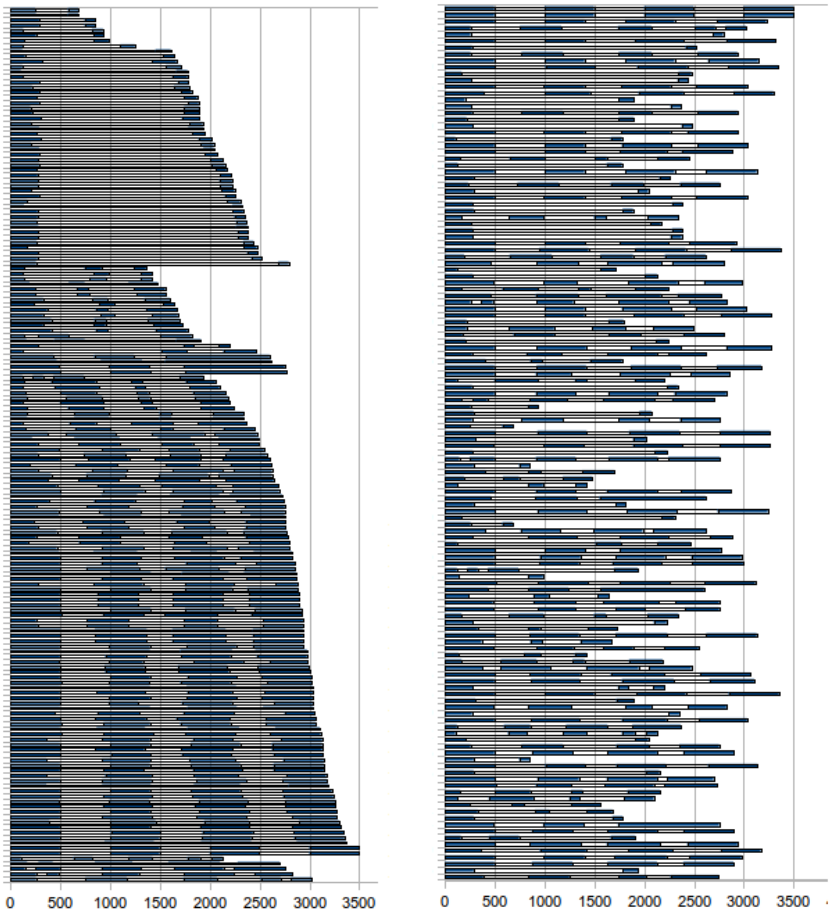


Fig. 3. Memes in a single robot's memory. Left: sorted by structure; Right: sorted by time (truncated). X-axis is the length of tune in milliseconds; Y axis – running order of the tune in dataset. Every bar represents an individual tune (meme), with dark grey areas indicating sound pulses and light grey areas indicating pauses.

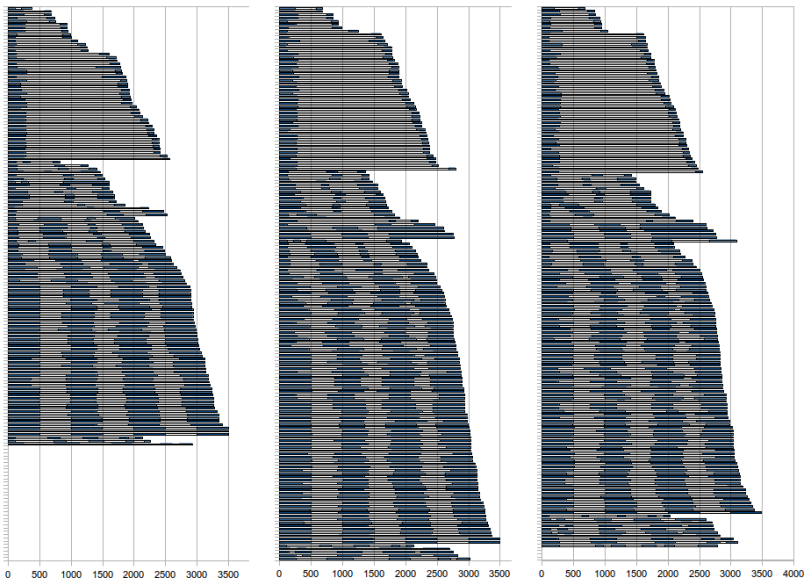


Fig. 4. Memes in memory of three robots, ordered by structure. X-axis is the length of tune in milliseconds; Y axis – running order of the tune in dataset. Every bar represents a tune, with darker areas representing sound, and lighter representing pauses. The robots were aligned in a row. The effects of distance/frequency-dependent attenuation can be seen in the difference between left and right charts.

3.4 The Critique

These individual-scale visualizations are straightforward representations of a small subset of experimental data. As such we are able to answer the question set. However, this visualization offers only a limited view of the processes that take place in the experiment.

We are able to identify clusters of memes, as well as several outlier memes, through a process of sorting based on meme characteristics (number of pulses and length). In doing this clustering we strip the temporal component out of the data. In the sorted data we observe a trend of smoothly increasing/decreasing meme length that is misleading – this is not present when we view the data in time series. Of more value is the use of these meme plots for 'odd-one-out' tasks as shown in Figure 4. It is clear that the robot on the left has a very different set of memes in memory because of its distance from the other two robots.

We only observe visual clusters and outliers when we sort the data to promote such trends; the side effect is that we create a new trend. The main unexpected insight from this visualization is that broadly two new forms of meme are generated, and these are readily observed in the sorted meme list. First, on occasion a pulse is not heard and so memes reduce in both length and number of pulses. Second, and unexpectedly, memes with a short pulse at either end of a very long silence arise from the asynchrony in the inter-robot communication.

This visualization will not scale well: it is a useful view of only a small number of robots. With a large population of individuals it is difficult for us to conceive of a visualization that would offer this detailed view. It also, clearly, reveals nothing as to the source of the dynamics: e.g. not representing the transmission data events from other robots. It is therefore not possible to interpret this dynamic without recourse to inference from the experimental design and comparative views of other individuals' memories. Of course, we could aggregate memes into groups of sufficiently similar memes and visualize a reduced level of detail but this may average over important novelty.

4 Community-scale Diversity

4.1 The Question

Which memory strategies promote or inhibit meme diversity in a robot community?

This stage of analysis compared the results of running simulations with different parameters, such as number of robots, speed of movement, initial memes, selection strategy, etc. We analyse the set of the memes generated by the community of robots as a whole, to enable bulk-scale comparisons of meme diversity under different experimental configurations, e.g., meme selection strategy. This may allow identification of patterns in the diversity of memes in the community meme-space. Here we considered communities of eight robots with four initial seed memes in each robot and two memory strategies. One memory strategy is the random pick as described above. The other memory strategy is a form of proto-imitation [2], where the e-Puck compares the memes it has just heard to the memes in its memory, determines which heard meme is most similar to one of the memes already in memory and sings the known meme from its memory. A distinction can be made between mimicry, the copying of actions, and imitation, recognizing the intent of those actions and attempting to achieve the same intent. With this proto-imitation strategy, the e-Puck determines the meme it believes the singer was trying to sing and sings that meme in response.

4.2 The Visualization

The set of memes recorded over several simulation runs was visualized as a three-dimensional scatter-plot (see Figures 5 and 6). The axes were three measures of a meme: x-axis is total meme length, in milliseconds; z-axis is number of sound pulses in the meme; and y-axis is a measure of the structural difference between the meme and an idealized meme of the same length and number of pulses. In an idealized meme all pulses are distributed evenly across the meme and are all of equal length. This structural measure is the log of the square root of the sum of squares of the differences in pulses lengths between that meme and its idealized form. Two memes are identical if they have the same total length, number of sound pulses and structure. Differences between memes may be manifest in any single metric or combination of metrics. We have no reason to favour any one of these metrics over another; consequently, we give equal weighting to differences in each metric. For visualisation, we plot these in Euclidean space and so as a result meme similarity is directly proportional to the Euclidean distance in this three-dimensional metric space.

4.3 The Interpretation

Visually, the overall structure can be seen as comprising several vertical strands and a fuzzy cloud at the top spreading along the shape defined by the columns. This structure presents the development of memes from pre-seeded ones, at the base of some columns. Seed memes lie at the base of columns since they have idealized structure. The columns are formed from variations on these memes that are still recognizably close to the original seed. The meme cloud at the top is the tunes that developed from pre-seeded ones but cannot be traced to the original. Of special theoretical interest is the cut-off point on a vertical axis after which the columns dissolve into the cloud, and cluster analysis (Figure 5) overlaid onto the visualization reveals this.

The volume of this cloud is sensitive to the meme selection strategy. Figure 7 shows a conservative proto-imitation strategy. There are fewer memes produced overall (number of points in the cloud) and fewer columns representing repeatable, distinct ideal memes than in the random pick strategy (Figure 5). This is expected since the proto-imitation strategy only imitates memes already known to it. Consequently, while memes heard may vary substantially from the ones in current memory, these are not sung and this limits variation.

In the sing random meme, there is much more variation in the meme cloud, and this is expected. What was not expected was the increase in

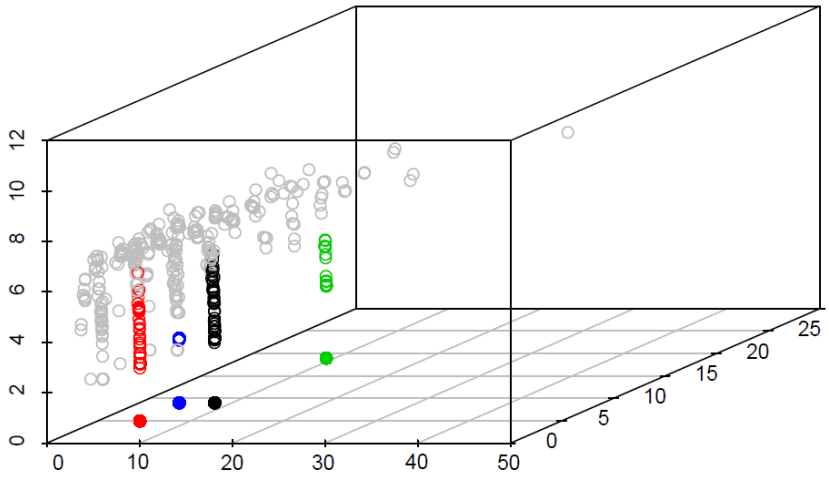


Fig. 5. 3D scatter plot of meme space, with random pick selection strategy. X-axis is length (ms); y-axis is a measure of structural deviation from an idealized meme; z-axis is the number of pulses. Red, Blue, Black and Green dots are memes that are close enough to be considered a variation of a corresponding seed meme (shown as solid circles).

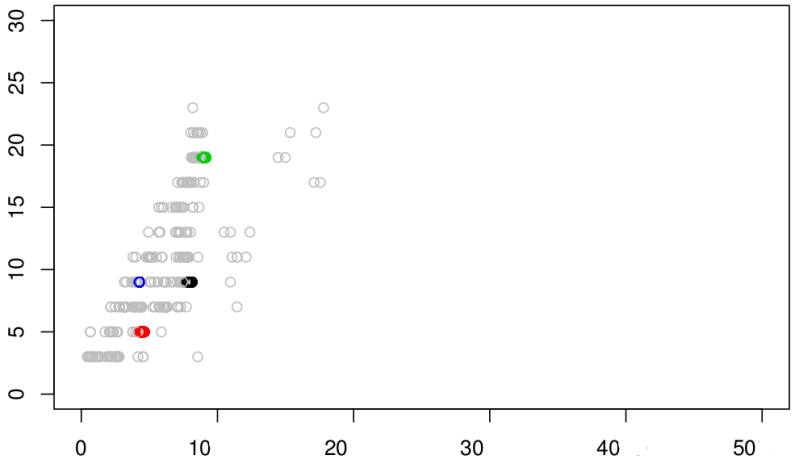


Fig. 6. 2D scatter plot of meme space, with random pick selection strategy. X-axis is length (ms); y-axis is the number of pulses. Red, Blue, Black and Green dots are memes that are close enough to be considered a variation of a corresponding seed meme (not highlighted in this figure).

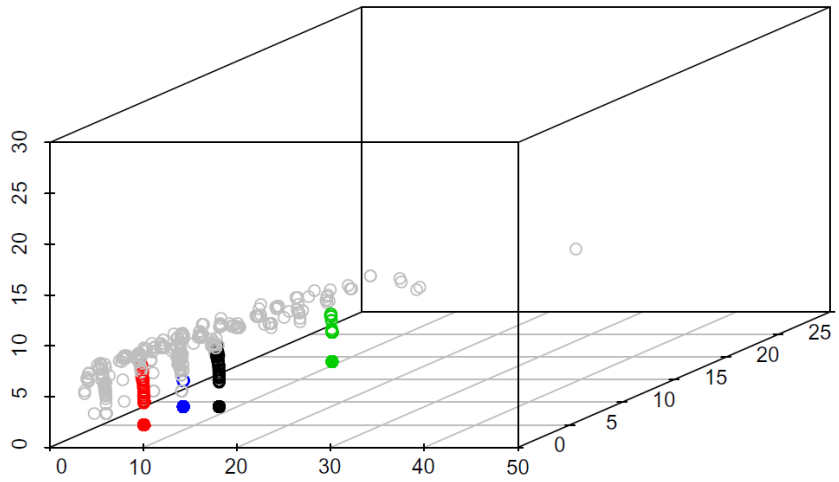


Fig. 7. 3D scatter plot of meme space, with proto-imitation selection strategy. X-axis is length (ms); y-axis is a measure of structural deviation from an idealized meme; z-axis is the number of pulses. Red, Blue, Black and Green dots are memes that are close enough to be considered a variation of a corresponding seed meme (shown as solid circles).

the number of vertical columns and the extending of the meme cloud along the x- and z-axes. A whole pulse being dropped from the original seed meme set causes the increase in number of vertical columns. Thus, new 'idealized' memes of unintended length are formed in the meme space. (Note, in the meme plot in Figure 5 this has occurred once but is far more common in Figure 7). The extension of the meme cloud along the x- and z-axes arises from meme concatenation: imitated memes overlapping in time and a single listener combining these two (or more) memes into a single meme.

4.4 The Critique

We get a clear view of the emergence of structure in the community of memes from the original (four) seed memes at the base of the diagram. The diagram also reveals clusters of repeated memes with a very small amount of noise, towards the meme cloud of variation. We can also clearly see the impact of selection strategy on diversity from the size and shape of this meme cluster. A new insight of very long memes with many pulses and silences is depicted as outliers (shown in the top right of these visualizations).

This form of diagram will scale – up to a visual saturation of points in the space – as simulation size increases and there is no time component. Of course, it is entirely dependent on our choice of axes, but nonetheless this visualization works well, except for the following caveats.

This visualization showed similarities among memes and hints at gradual evolution of memes along vertical paths in the meme space, from the pre-seeded memes into the cloud at the top. However, this impression of smooth trends in meme evolution may be misleading, as was seen in the memory visualization of the single robot case. Here, we are not able to see how this meme space evolved over time – we only observe it at the beginning (implicitly through seed memes) and at the end (explicitly). Added to this, we have no sense of the degree of repetition in this meme space.

5 Meme Propagation

5.1 The Question

How do memes propagate over the robot community, and how does meme memory strategy impact this?

To explore progression from seed memes through to memes that are more varied, and especially meme reproducibility, i.e., how many repetitions of a given meme exist, a visualisation was developed that focuses on meme imitation events. We considered two memory selection strategies: distinct and grouped. A robot with distinct memory will store every meme heard as a new meme, even if it is identical to a meme already in memory. A robot with grouped memory will examine every meme heard and determine if it is already known, in accordance with a similarity threshold with respect to the length/pulses/structural metric described above, or new. New memes are added to memory, already known memes have their count incremented.

5.2 The Visualization

We show data resulting from a single simulation run with eight robots. A data unit is a single interaction: a robot singing a meme, and a robot (not necessarily a different one) hearing it and imitating what it heard (although not necessarily the same meme since errors may be introduced as described above). The parameters of an interaction were: the identity of the originating and the imitating robots; and the memes played by each of them. Some of the memes were classified as seed memes, being either identical to original memes or sufficiently close to them. Other

memes were classified as novel, based on the metric developed in the previous analysis.

These data were visualized as a link-chart, which we call a memeograph, where every node represented a robot meme, and every edge is a listening event. The spatial layout of the nodes and the routing of the links was undertaken using the hierarchical layout algorithm, implemented as a part of Graphviz software package (www.graphviz.org). The shape of a node glyph corresponded to the robot identity, and the combination of two colours (fill and border) represented meme identity. Glyph size represented meme novelty: large glyphs represented preseeded memes, and small ones – novel memes.

5.3 The Interpretation

Figure 8 shows the impact of the direct memory strategy, and highlights a number of features of meme evolution. First, there are long (vertical) chains of the same glyph: catchy tunes. These are selected at random but are repeated with increasing frequency and by an increasing number of robots (i.e., different shapes) as the simulation progresses. For example, see the long run of dark squares (a particular meme sung by the square robot), which is repeated frequently by other robots as well (circle, triangle, hexagon etc.). This particular seed meme pervades the length of the memeograph.

There are other examples of non-seed memes exhibiting this same pervasiveness. The right-hand branch of Figure 8 shows three different newly generated memes, depicted as small shapes, occurring over several imitations. However, there are just as many non-seed (small) memes that fail to catch on. These are depicted by small memes that exist at the terminal end of the sequence.

For the grouped memory strategy, the overall memeograph topology is characteristically different (see Figure 10). It necessarily has fewer nodes – because of the grouping. Edges are further annotated by the number of times they are repeated. It may be observed that the same dark seed meme is repeated many times, depicted by the 'pig's ear' arrow on several different robots (different shapes). Likewise, there are several non-seed memes with a similar number of repetitions. Indeed, as with the direct memory strategy, there are many occasions of seed memes and new memes failing to be repeated at all.

5.4 The Critique

This is a complex visualization for a complex transmission dynamic. We have had some success in answering the question set, in that we can

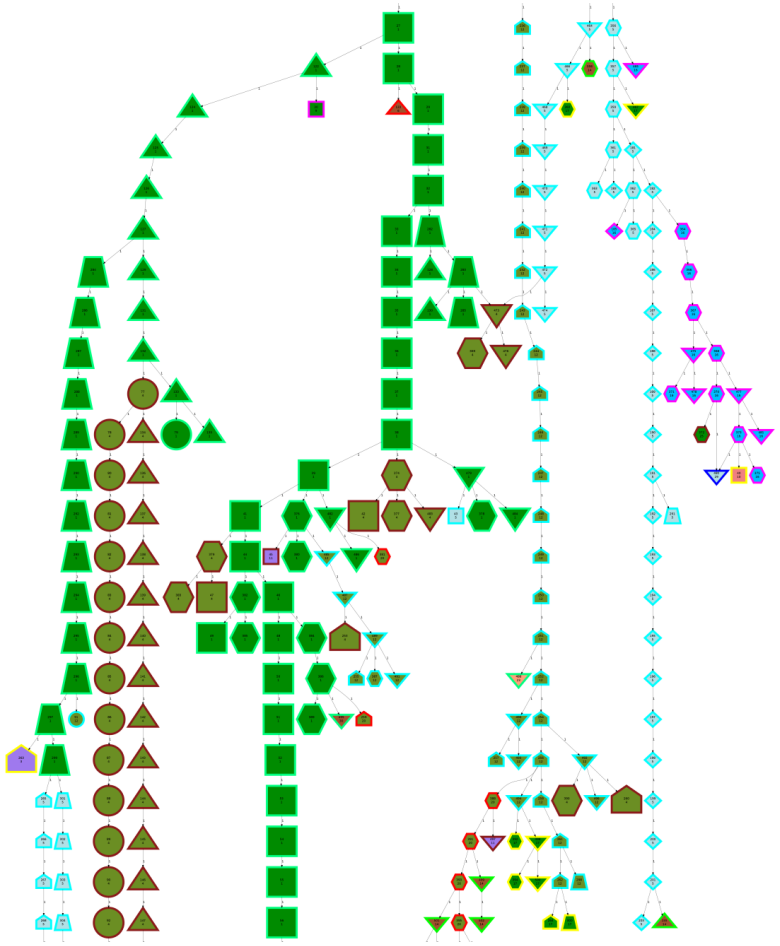


Fig. 8. Memeograph for direct memory strategy, showing long chains of meme repetitions for some memes and other memes that are imitated once and not repeated. Each node represents an instance of a meme in the memory of a single e-Puck. The shape of the node identifies the e-Puck possessing the meme. The meme is identified by the colour combination of the node. Large nodes are seed memes; small nodes are new memes.

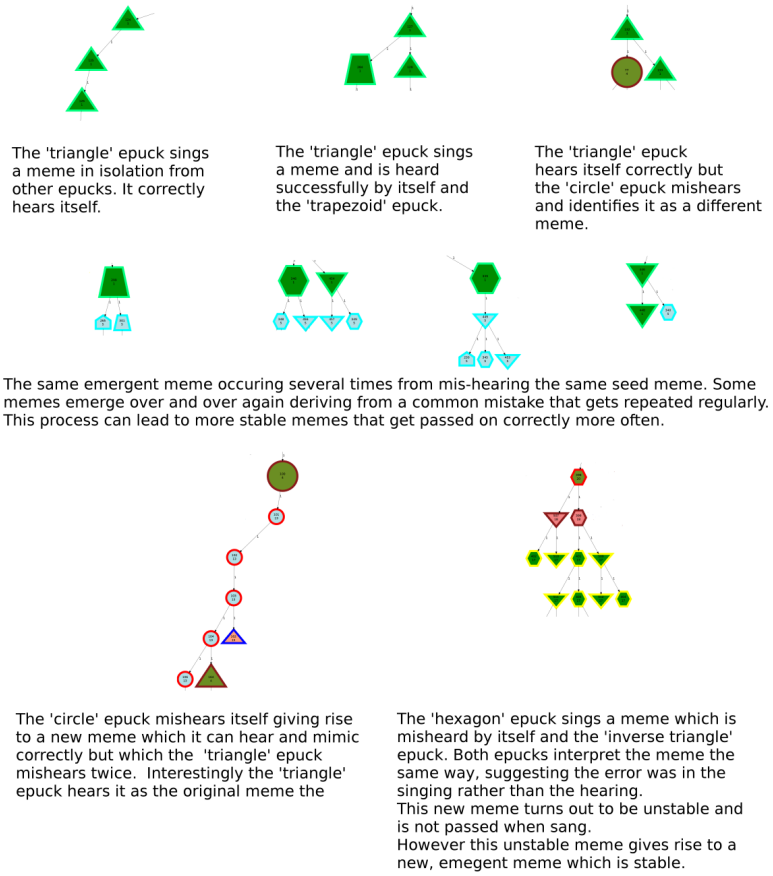


Fig. 9. Explanation of details of memeographs, examples taken from Figure 8.

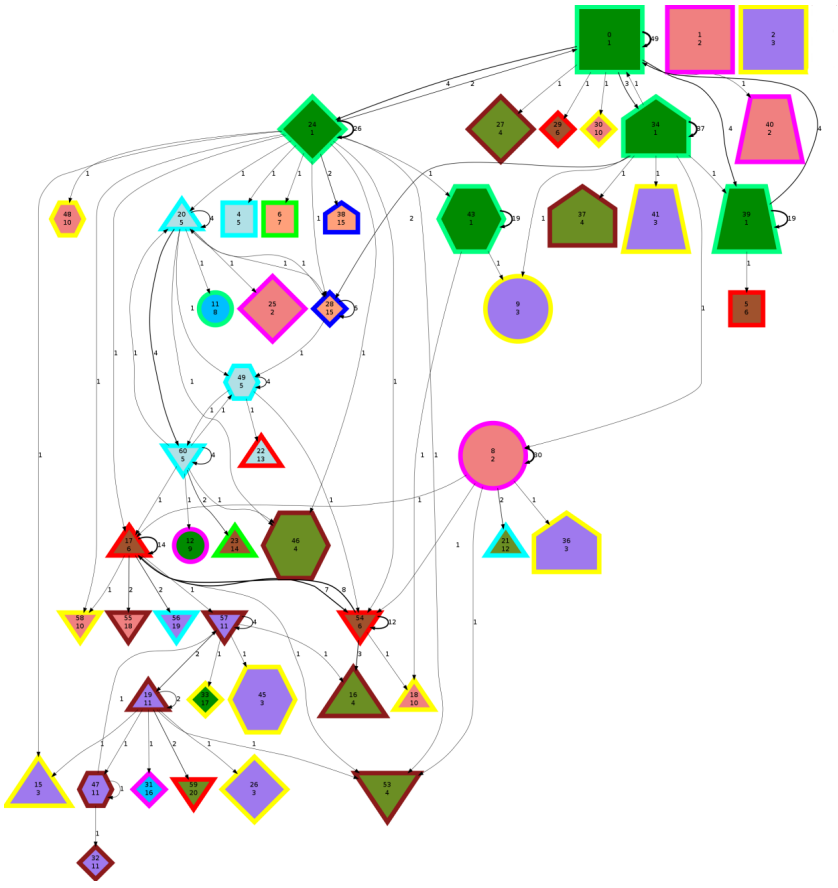


Fig. 10. Memeograph for grouped memory strategy, with lines showing both the repeated memes and meme variations as a consequence of transmission.

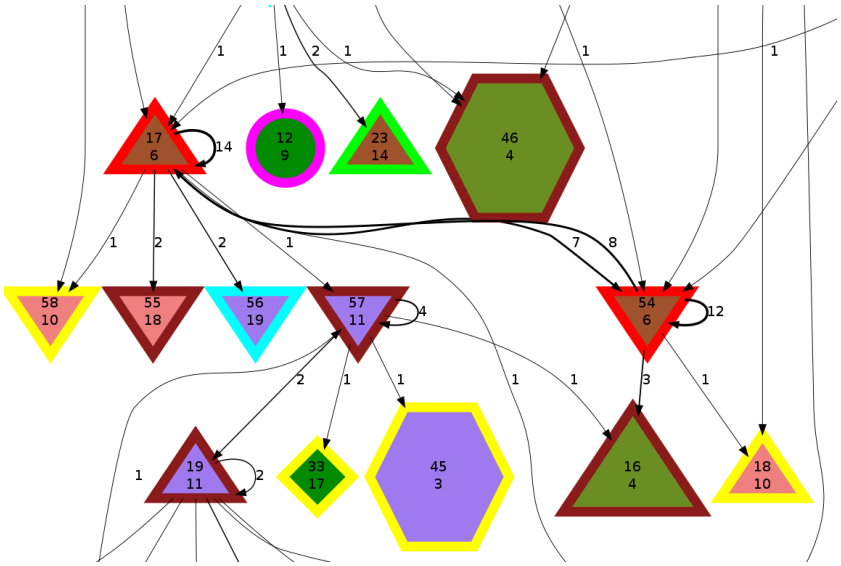


Fig. 11. Memeograph for grouped memory strategy, with lines showing both the repeated memes and meme variations as a consequence of transmission. Detail view of part of the memeograph from Figure 8. The thickness of the links and the numbers on the links indicate how many times the tail meme has been interpreted as the head meme.

see the impact of memory strategy and selection strategy (not shown) on memeograph topology. We make use of trend visuals to depict long trails of repeated memes and readily spot outliers – i.e. branches in these sequences and single node sequences. However, caution must be taken when interpreting the trends, since it is tempting to imagine this is time series data. A given node-edge-(same colour) node link represents a single event of a meme being repeated. Several time steps may have passed between each imitation and the imitating robot may have repeated other memes in between. We do not make use of any clustering in these mimeographs, aside from the implicit clustering of the grouped memory strategy.

This is the first time we have attempted such a visualization, and there are many features that could be improved. In this visualization, shape is related to robot, and size and colour to meme. Perceptually, shape and size are automatically aggregated into object identity. Grouping size and colour and dissociating these from shape is perceptually

unnatural. This means that we make associations more easily between two unrelated visual features and provide an unintuitive view of the data.

Our visualization also makes poor use of colour (not shown here but available at http://sites.google.com/site/artcultproject/). The different meme identities, shown as colour combinations, give no indication to meme similarity: specifically, the allocation of colours to memes is entirely at random. Correlating meme similarity with hue similarity would have provided us with a mechanism to depict meme similarity, and this would have answered our question on whether memes changes gradually or suddenly over imitation events. In its present form, it is not possible to determine this. The numbers on edges and inside glyphs are not easily readable. The range in variation of edge thickness (1 point to 3 points) does not allow for high-fidelity representation of the range of values in the data. A better option would be to increase the thickness of all edges to the maximum (here 3 points) and map the data onto the edge colour intensity. The visualization provides no information on the closeness of robots during transmission. Given the shape of the sound attenuation curve (above) this would not be difficult to factor in – a distinction between immediate vicinity and any further away (but still able to hear the transmission) would be sufficient. Therefore, it is not possible determine the relative contribution of meme reproduction error and distance to variation.

6 Discussion

We appraised three visualizations – each at different scales – in our artificial culture system. We considered this appraisal in terms of the commonly performed visual tasks of detection of outliers, trends and clusters. When mapping these typical visual search tasks back into the data domain, outliers and clusters were shown to reveal new insights in the robot community dynamics. For example, in the robot memory visualization (Section 2) we observed as outliers an unexpected meme comprising two short pulses punctuated by a very long silence. We also observed new, very short memes. Both of these features could be explained in terms of the underlying robot-to-robot interactions, but without visualisation it would have been difficult to determine that this dynamic occurred.

We also observed individual variation among robots. By visualizing the internal memory of three separate robots, we could immediately see the impact of spatial arrangement on memes stored (Figure 4). Two robots were very similar in this regard; the robot furthest away from the robot with the weak voice was characteristically different. Moreover, analysis of these single-robot scale visualizations directly informed devel-

opment of metrics which were used in subsequent larger scale analyses. Specifically, we could see the changes in structure that could arise from robot-to-robot interactions, and this was not possible to determine in advance of the simulation.

Likewise in the community scale visualizations (Section 3), we noted concatenated and truncated memes. Similarly, these phenomena are explained by the asynchrony in robot communication. Visual clustering revealed consistencies and differences in both of these visualizations. For example in both Figures 5 and 7 we could clearly observe seed memes, copies (or sufficiently close copies) of the seed memes and the new memes arising from a mix of errors in meme transmission and differences in meme selection strategies. In particular, the impact of meme selection strategy on community diversity was obvious from visual inspection of the clusters in Figures 5 and 7.

Observation of trends typically stimulated new enquiries: a suggested trend of a smooth meme lengthening / shortening process over time was readily dismissed (Figure 3 (left)) with a different visualization (Figure 3 (right)). Discussions between the primary researcher and the external party revealed an inferred trend of a smooth evolution from idealized memes into a diverse cloud over time (Figures 5 and 7). We sought to investigate this trend with our most complicated and new, at least to us, visualization of the memeograph.

Perhaps unsurprisingly, this unconventional visualization attracted the most critique.

In this visualization, we made poor use of the mix of size, shape and colour. We bound together colour and size for memes and shape for robot, yet in the perceptual system size and shape are readily aggregated and this ready aggregation should be recognized and factored into the design. Further, colour could have been used to understand the nature of the evolution of memes in the community, by linking similarity in colour to similarity in meme (based on our meme metric). We will update the visualization accordingly in subsequent analyses. Other refinements suggested above will also be factored into our visualization approach.

We are not able to derive easily generalities or repeatability from our memeographs. Memeographs are too detailed in their representation of individual interaction. While it is possible to see small-scale details of mutations of a specific meme, no overall characteristics of a simulation (except the overall shape of the memeograph network) can be derived. General characteristics relating to meme diversity are better represented by the 3D scatterplot. Regarding repeatability, a specific simulation may be run a number of times resulting in a number of very different memeographs, while the 3d scatterplots are consistent,

the structure of the 3d plot revealing common features of the nature of that specific community. The 3d plot retains details of the length, timing and structure of the memes which the memeograph does not.

More generally, a given visualization construction can only properly answer the question it was designed for. Trends, outliers and clusters may emerge, as we have shown, that suggest additional interpretations to the data. While these are useful for stimulating different, new questions it is not enough to rely on the existing visualization to answer new questions. In this respect, it is like any other model, e.g., a simulation model. It is fit for the purpose it is designed for, and extending the purpose without extending/ reviewing the design is just as inappropriate.

With regard to scaling, we have presented three different visualizations at three different scales. Only the community visualization will scale to communities with much larger numbers of agents. This scaling can continue up to the loss of clarity in the 3D plot, which to some extent may be alleviated through interactivity in 3D rotational navigation. Clearly, the robot memory and robot-to-robot communication memeograph visualizations scale poorly with the number of robots, although the robot-to-robot communication will scale better. One solution is to generate more abstracted and coarse-grained representations of each – e.g., collapsing the meme list down through grouping and then storing frequency of memes for each group for robot memory – would help address the volume of data introduced by up-scaling. Another solution is to adopt an interactive overview and zoom visualisation approach where this coarse-grained overview may be shown for selected sub-populations of robots, and there is always the opportunity to probe in detail individual robots located in space over time.

More interestingly, visualization itself may offer a pattern-based solution to dealing with the increase in data that is an obvious consequence of an increase in scale. Note that our third visualization (4) arose out of a need to understand the relation between individual scale exchange of memes – the process – and community-scale meme evolution – the pattern.

We propose that an effective (improved) visualization of robot-to-robot communication that shows the similarity of memes involved in transmission events can help link from process to pattern and deal with an increase in spatial scale. Specifically the list of memes imitated (and that could be heard) at the time of listening and the resulting meme stored in memory, may be useful in linking scales. Such a visualization, as a precursor to more rigorous statistical methods, would enable derivation of a probability density distribution of the meme transmission process from robot-to-robot. This probability density function would be

parametrized by local neighbourhood variables such as the number of other robots in the vicinity and a frequency distribution of the meme memories (as described above) in neighbourhood. In this way, visualization may act as a guide to simulation reformulation into higher-level, abstracted processes – ultimately founded on statistical averaging – that build in links in scales and ease the up-scaling process.

Acknowledgements

This work is supported by the UK Engineering and Physical Sciences Research Council (EPSRC) grant reference EP/E062083/1 and the authors gratefully acknowledge project co-investigators and researchers.

References

[1] N. Andrienko and E. G. Andrienko. *Exploratory analysis of spatial and temporal data: a systematic approach.* Springer-Verlag, 2006.

[2] P. Andry, P. Gaussier, and J Nadel. Simulations of dynamical interactions for social learning. In *European Workshop on Learning Robots, EWLR01*, pages 57–64, 2000.

[3] J. Bertin. *Smiologie graphique.* Paris: Mouton, 1967.

[4] J.L. Bown, E. Pachepsky, E. Eberst, U. Bausenwein, P. Millard, G.R. Squire, and J.W. Crawford. Consequences of intraspecific variation for the structure and function of ecological communities part 1: Model development and predicted patterns of diversity. *Ecological Modelling*, 207:264–276, 2007.

[5] R. Dawkins. *The Selfish Gene.* Oxford University Press, 1976.

[6] R.E. Falconer, J.L. Bown, N.A. White, and J.W. Crawford. Biomass recycling and the origin of phenotype in fungal mycelia. *Proceedings of the Royal Society B*, 272:1727–1734, 2005.

[7] P. Humphreys. *Extending Ourselves: Computational Science, Empiricism, and Scientific Method.* Oxford University Press, 2007.

[8] D. A. Keim, F. Mansmann, J. Schneidewind, and H. Ziegler. Challenges in visual data analysis. In *Tenth International Conference on Information Visualization*, pages 9–16, 2006.

[9] J. Mackinlay. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics (TOG)*, 5(2):110–141, 1986.

[10] F. Mondada, F. Bonami, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, Zufferey J., D. Floreano, and A. Martinoli. The e-Puck, a robot designed for education in engineering. In *9th Conference on Autonomous Robot Systems and Competitions*, pages 56–65, 2009.

[11] W. Playfair. *The Commercial and Political Atlas.* 1801.

[12] M. Prokopenko, F. Boschetti, and A.J. Ryan. An information-theoretic primer on complexity, self-organisation and emergence. *Complexity*, 15(1):11–28, 2009.

[13] J. Redish. Expanding usability testing to evaluate complex systems. *Journal of Usability Studies*, 2(3):102–111, 2007.

[14] M. M. Shovman, A. Szymkowiak, J. L. Bown, and K. C. Scott-Brown. Use of 'pop-out' paradigm to test graph comprehension in a three-dimensional scatter plot. *Perception*, 37 (ECVP Abstract Supplement):79, 2008.

[15] M. M. Shovman, A. Szymkowiak, J. L. Bown, and K. C. Scott-Brown. Changing the view: towards the theory of visualisation comprehension. In *IEEE conference on Information Visualisation, Barcelona*, 2009.

[16] J. J. Thomas and K. A. Cook. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE Computer Society Press, 2005.

[17] E. R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 1986.

[18] C. Ware. *Information Visualization: Perception for Design*. Morgan Kaufman, 2000.

[19] A.F.T. Winfield and M. Erbas. On embodied memetic evolution and the emergence of behavioural traditions in robots. *Journal of Memetic Computing*, (to appear), 2011.

[20] A.F.T. Winfield and F. Griffiths. Towards the emergence of artificial culture in collective robot systems. In P. Levi and S. Kernbach, editors, *Symbiotic Multi-robot Organisms*, pages 431–439. Springer-Verlag, 2010.

Multiple model simulation: modelling cell division and differentiation in the prostate

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

YCCSA, University of York, UK, YO10 5DD
Alastair.Droop|Fiona.Polack|Susan.Stepney@york.ac.uk;
Philip.Garnett@yccsa.org

Abstract. We describe an approach to building a hybrid multi-scale model, using a Petri net model for the top layer, and object-oriented models at the lower layers, with a rigorous definition of how the layers compose. We apply this approach to building a model of prostate cell division and differentiation, with each model layer describing the processes at a suitable level of abstraction. This provides a systematic modular approach to modelling and to model validation, allowing use of diverse modelling techniques in developing multi-scale models with no loss of rigour or validity. We use this prostate model to develop a prototype simulation, and demonstrate that it is capable of simulating biologically-relevant numbers of cells.

Keywords: complex systems, agent based modelling, Petri nets, object models, prostate

1 Introduction

Simulations of complex systems have the potential to complement laboratory based biological research, providing a platform for repeatable experimentation, and for exploration of parts of biological systems that cannot be analysed directly in live organisms or dead tissue. Elsewhere [1, 11, 33, 34, 35], we consider how scientific simulations can be designed and validated.

In this paper we present the modelling and simulation of cell division and differentiation in the prostate. The results of this simulation will be used as hypotheses to guide laboratory experimentation by suggesting the cellular behaviours and pathways which are most likely to be involved in early cancer neogenesis.

Prostate cancer is a common disease, accounting for almost 25% of male cancers in the UK. As with many cancers, prostate cancer is a complex disease arising from the aberrant behaviour of a tissue. High-grade prostate cancer is characterised by a breakdown of normal cell differentiation behaviour, leading to a proliferation of terminally-differentiated cells. The study of prostate cancer therefore requires an understanding of the dynamics of a large population of cells. Many of the genomic events that lead to cancer formation are rare, stochastic events. As such, a continuous modelling approach (for example based upon ordinary differential equations) is unable to capture the rich behaviour necessary to model cancer neogenesis. The construction, parameterisation and analysis of the complicated models needed to study prostate cancer requires input from multiple domain experts. The work reported here has been the result of close collaboration between domain experts in the YCR Cancer Research laboratory and software developers in the York Centre for Complex Systems Analysis (YCCSA).

The prostate domain biology provides insight into cell transitions and differentiation, transition rates and proportions of each cell type that can be expected in normal, modified and cancerous prostate. However, as in all systems biology, there are significant gaps in the domain knowledge, and there are many features that cannot be accurately studied. In simulating the prostate cells, we seek to be able to (a) replicate the cell behaviours and proportions in a normal prostate; (b) replicate the modifications that biologists make – for example, knock-out experiments to determine causality – and to show that appropriate behaviours are simulated in the modified-prostate simulation; (c) to experiment with the various forms of point mutation that have been postulated as initiators of prostate cancer. The need to precisely measure rates and proportions, and, in particular, the need to accurately express and explore interventions and point mutations, dictates the use of an individual-based model; we use an agent-based simulation. In modelling, we use diagrammatic approaches, which are easy to explain to domain experts, as well as being familiar software engineering tools.

For the simulation, key features are the model of state change (differentiation) and cell creation (division) in the prostate. We need design notations that help us in validating the domain and conceptual models with our domain experts. These models must also admit continuity in development, and be verifiable. We also need to view the development process as part of the construction of an argument of validity, which requires us to systematically identify and record assumptions.

We follow the CoSMoS process: a principled, systematic development, in which domain experts co-operate closely with simulation developers.

A *domain model* captures the domain experts' view of the part of the system to be simulated, and the desired scope and purpose of simulation. From the domain model, a *platform model*, suitable for simulation development, is produced. The *simulation platform* is then constructed using a systematic software engineering process. Validation is emphasised, both in relation to the biological bases of the simulation and the design decisions of the software engineering.

In [35] we note that the validation of the implemented simulation is facilitated if the software engineering models use modelling approaches that map seamlessly with the implementation platform and language(s). In simulating aspects of prostate cell processes, we demonstrate that appropriate modelling not only facilitates software engineering and validation, but can also facilitate modelling and validation of the biology underpinning the simulation and simulation experimentation.

1.1 Modelling notations for systems biology

Petri nets are widely used in modelling biological systems. Petri nets (see §2) are based on state-transition diagrams and the theory of automata, they have strong mathematical underpinnings, and can provide formal verification of properties. However, in computational modelling of biological systems most uses encounter the limitations of reactive-systems modelling. For example, for the modelling of biological pathways, it has been found necessary to extend Petri net notations with continuous features: Matsuno et al propose, and extensively use, *Hybrid Functional Petri Net with extensions* to accommodate continuous places and transitions [28, 31]. Such adaptations of Petri nets have the advantage (usually) of maintaining the strong formal basis. However, the resultant models are poorly structured and hard to read – this inhibits validation of models against the biology. The scale and density of the Petri net models also makes them hard to implement in current computer languages.

In 1987, Harel [14] noted the inappropriateness of conventional state-transition modelling for transformational systems. His widely-adopted answer is statechart diagrams, "a visual formalism for describing states and transitions in a modular fashion, enabling clustering, orthogonality (that is, concurrency) and refinement, and encouraging 'zoom' capabilities for moving easily back and forth between levels of abstraction." [14, p233]. Statechart-like notations are readily understood by many biologists [12], and have been used in biological modelling [22, 43]. However, statechart modelling also suffers from aspects of inadequacy. In systems with significant state (those where data are generated, stored and used, and values persist over time), statecharts cannot capture data structures

or transactional (and reactive) structures adequately. Like Petri nets, attempting to use a single statechart to capture a heterogeneous biological system results in a complicated, dense model which is hard to validate and to implement from.

Some biological modellers turn to UML-like modelling notations. UML (Unified Modeling Language) [32] has many attractions: it provides notations for modelling static and transactional structures as well as statechart-like notations; there is conceptual unification across notations (through a metamodel-based language definition); there are lots of tools; some of the notations map readily to common programming languages (principally, Java). However, UML is a unified *object-oriented (OO) modelling* language, and the connotations of object-orientation are not always appropriate for biological systems. The most significant issues for modelling cell division and differentiation are that objects cannot reproduce and divide, and an object cannot change its type (class). Continuous concepts such as cell populations and concentrations do not map well to the instance-based OO modelling concepts (class, object, slot, value). These problems are usually ignored in biological modelling [11, 12, 38, 39, 40, 48, 49]: it is considered sufficient that the diagrams say something about the biological structures, and map to Java.

To summarise: Petri nets are a clear, elegant way to model reactive systems; statecharts allow clarity in the modelling of transformational systems; OO modelling notations such as UML provide useful and complementary visuals if we can ignore issues of semantics.

1.2 Hybrid models

Most modellers use one base notation and extend it, adapt it or abuse it to model the whole of a biological system. However, this approach loses much of the clarity that abstract modelling should bring to a problem.

There is some multi-notation modelling in biology. Harel et al [7, 8, 15, 16] address the inadequacies of statecharts for biological modelling in Reactive Animation, a proprietary modelling framework that uses statecharts for transformational aspects, object models for static structure, and live sequence charts for transaction structures. The notations are integrated through their roles in generating and maintaining a simulation implementation. Setty, Cohen, and Harel [16, 42] use Reactive Animation to model pancreatic organogenesis. Statecharts are used to model the individual cell lifecycle; other models capture the morphology of the system. The cell division process is captured only informally: "Proliferation ends when the `Cell` duplicates itself by creating an identical instance. In turn, a message is sent to the front end, which creates a new identical sphere corresponding to the new `Cell` at the proper location."

[16, p9]. In Reactive Animation, models are complementary, and the aim is to progress as far as possible in the analysis *without* connecting the models [42].

The hybrid modelling approach that we use is not proprietary (or tied to the modelling notations used here): it focuses on finding an appropriate notation for each aspect of the system, and an appropriate way to integrate the component models, for example through input-output mappings. Whilst we anticipated that this would cause problems in implementation, it turns out to be a natural way to construct and structure simulation code. Appropriateness is determined primarily by the ability of the notation to express domain concepts in ways that the domain experts, as well as the simulation developer, can understand. However, it is also important that models map clearly to code concepts, to support software validation.

1.3 Structure of this paper

In this paper, we use multiple models, apply them to modelling cell differentiation in the prostate, and reflect on the advantages, opportunities and limitations of such an approach. §2 introduces Petri nets and Petri net terminology, and introduces a high-level model of cell differentiation and division. §3 introduces three notations (loosely based on UML) that are used to model the lower-level individual cell behaviours. The behaviour of a cell is determined by its place in the Petri net, and its environment. A key aspect is that, in moving between two places, a cell undergoes a change in its active behaviours, and its internal structure. §4 describes how the different models and layers can be combined. §5 discusses how our approach amounts to defining a domain specific language. §6 presents a prototype model of the prostate cell structure. §7 describes the simulation implementation: this prototype replicates the dynamic changes in cell-type proportions in a normal prostate. §8 outlines further work in developing the modelling approach, and in the prostate model itself. §9 presents our conclusions.

2 The high level system structure: Petri net modelling

The high level Petri net model captures the structure of the cell division and differentiation process, without considering the detailed internal structure of the component cells. Petri nets are widely used to model signalling and pathways in biological systems (see, for example, [5, 13, 27, 41]), and are attractive to many systems biologists. They

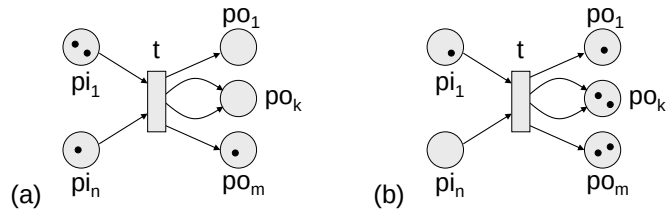


Fig. 1. Example Petri net. (a) A net with n places $\mathsf{pi}_1 \ldots \mathsf{pi}_n$, input to the transition t, which has outputs to m places $\mathsf{po}_1 \ldots \mathsf{po}_m$. Place po_k has two incoming arcs. The net's current marking is 2 tokens in place pi_1, one token in pi_n, and one token in place po_m. (b) The same net after transition t has fired. One token has been consumed from the place along each input arc, and one token has been produced in the place along each output arc. Hence two tokens have been produced in place po_k, one for each of its arcs.

provide a natural way to model cell division (one object becoming two objects) and cell differentiation (one object becoming a different type of object).

2.1 Petri net concepts and definitions

The (draft) Petri net ISO standard website [17, 18] defines a Petri net as "a formal, graphical, executable technique for the specification and analysis of concurrent, discrete-event dynamic systems". There are many flavours and varieties of Petri net; here we describe only the features necessary for our model. The interested reader is referred to the extensive bibliography accessible from the ISO standard website.

A Petri net is a bipartite directed graph. The two kinds of nodes are *places* and *transitions*. Place nodes have a *marking*: a set of *tokens* occupying the place (figure 1a). A transition can *fire* when all its input places hold a token: one token per input arc is *consumed* from its input place, and one token per output arc is *produced* in the respective output (figure 1b). Some descriptions instead talk of tokens flowing around the net; here we emphasise that tokens are consumed and produced by transitions, and so emphasise that their numbers and types need not be conserved, as the focus of our modelling is on the production of new cells and new kinds of cells.

When Petri net models get large, it can be helpful to structure them. Fusion places [19, ch.5], an extension to standard Petri nets, allow a Petri net to be presented in several pieces. A fusion place is marked on several nets, which means it should be identified across the nets (figure 2). We

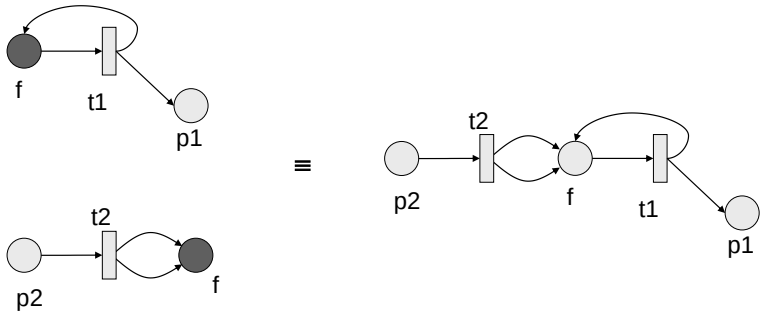


Fig. 2. Example fusion place. The two nets on the left have the fusion place f (dark grey). They are equivalent to the net on the right.

use fusion places to help structure the chain of cell differentiations. A fusion transition can be used to capture an entire sub-net; we could use a simple version to help capture the repeated pattern of cell divisions, but do not do so here (see §8.3).

2.2 Modelling cell interactions

In our Petri net model, each cell is modelled as a token, and each type of cell is modelled as a place. The way cells differentiate (change type) and divide is modelled by the transitions. We name all the places and transitions in the net; these names are used to link the Petri net model to the lower level models (§3).

Figure 3 shows a variety of differentiation and division processes modelled using a Petri net. These various possibilities can be captured in a single model of cell differentiation, figure 4. (Our actual model of prostate cell differentiation and division includes more detail; see §6.) There are two possible models. Figure 4a shows the case where the two daughter cells are distinguished separately. This allows asymmetric transition probabilities (with d1 being highly likely to remain a p1 cell, and d2 being highly likely to differentiate, for example). Figure 4b shows a simpler representation, that cannot capture the asymmetry at this level of modelling. As we include a lower layer of modelling, we use the simpler representation, and devolve handling the asymmetry to the lower level.

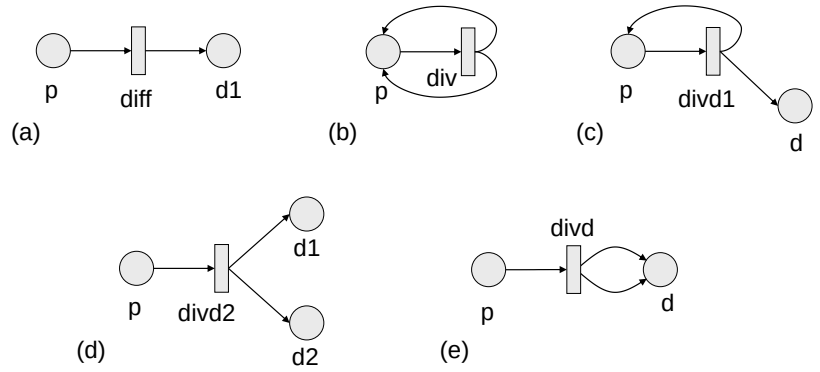


Fig. 3. Simple Petri net models of cell differentiation and division. (a) cell of type p differentiates into cell of type d1; (b) parent cell of type p divides into two daughter cells also of type p; (c) parent divides into two daughter cells, one also of type p, one differentiated into a daughter cell of type d; (d) parent divides into two cells, one differentiated into type d1, one into type d2. (e) parent divides into two cells, both differentiated into type d.

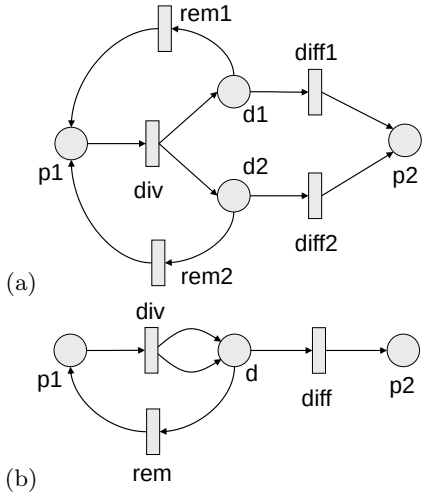


Fig. 4. A Petri net model of cell division incorporating all differentiation possibilities. (a) a cell p1 divides into two daughter cells d1 and d2, which each chose either to remain a p1 cell type (transition rem), or differentiate into a p2 cell type (transition diff). (b) a cell p1 divides into two daughter cells d, which each chose either to remain a p1 cell type (transition rem), or differentiate into a p2 cell type (transition diff).

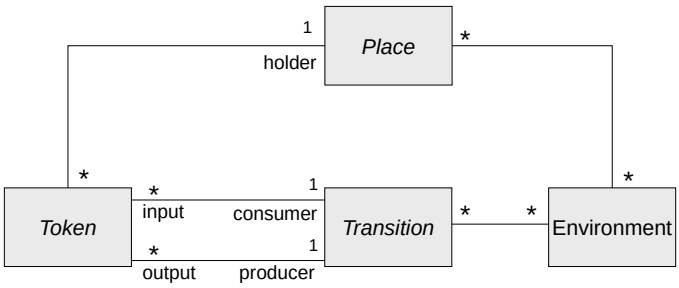


Fig. 5. The class model for the tokens. A token is in a place. Token and Place are abstract classes; there is one subclass of each for each Petri net place. A transition may consume many input tokens; a token is consumed by precisely one transition. The transition may produce many output tokens; a token is produced by precisely one transition. Transition is an abstract class; there is one subclass for each Petri net transition. Places and transitions may be associated with many environments, which can affect their behaviour.

3 The low level component structure

The low level component model captures the behaviour of an individual cell through its life cycle, and the details of the division process where one cell becomes two, and includes interactions with the environment. We use object-oriented modelling notations to capture this structure (see, for example, [9]).

3.1 Modelling cells: classes and objects

A cell, modelled as a Petri net token in the system layer, is modelled as an object (instance of a Token class) with state and behaviour in the component layer. Each place is modelled as a (singleton) instance of its Place class (every place is different), as it too can have state and behaviour. The behaviour of a transition is modelled by an instance of the Transition class, which *consumes* token objects from the input places, and *produces* token objects into the output places. We also allow the model to have environment(s) (instances of the Environment class or its subclasses), to provide inputs to the behaviours (for example, to the guards on state changes, or to the mutation rate on cell division). The relevant class model is shown in figure 5.

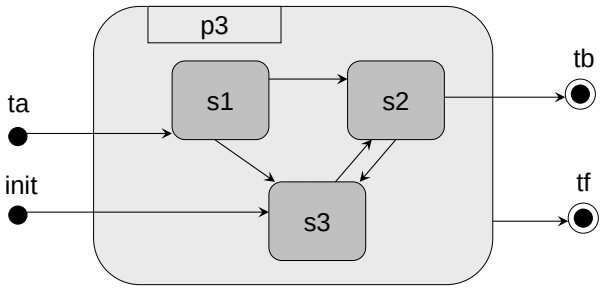


Fig. 6. An example state diagram. If the token in place `p3` in the Petri net layer is produced from transition `ta`, it starts in state `s1`; if produced externally, it starts in state `s3`. The cell can exit to transition `tb` only from state `s2`; it can exit to transition `tf` from any substate.

3.2 Modelling cell state: state diagrams

The behaviour of a token in a place is modelled using a state diagram; each place has a different kind of token (cell), and so has its own state diagram. This has one or more start states, labelled with a name (corresponding to a Petri net layer transition name, or to some initialisation), and one or more end states, also labelled with a transition name. For example, see figure 6.

3.3 Modelling cell transitions: sequence diagrams

In the Petri net layer, the transition is where a cell divides into two new cells, or differentiates into a new kind of cell. The low level modelling allows details of these processes to be specified, such as what information about the cell survives the transition, and what changes. We use a transient transition object for this, which 'consumes' the input cell, and 'produces' the new output cell(s). In some sense, this transition object can be thought of as 'being' a cell in its transition state of dividing or differentiating.

So, in object-oriented modelling terms, a transition object (figure 5) exists for the duration of the transition behaviour. It is initialised with any necessary information from the input token object(s) and the environment, performs appropriate behaviours, and initialises the output token object(s). The input tokens(s) terminate once the transition object is initialised. The transition object terminates once the output token(s) are initialised. The interactions can be modelled with a sequence diagram; a typical example is shown in figure 7.

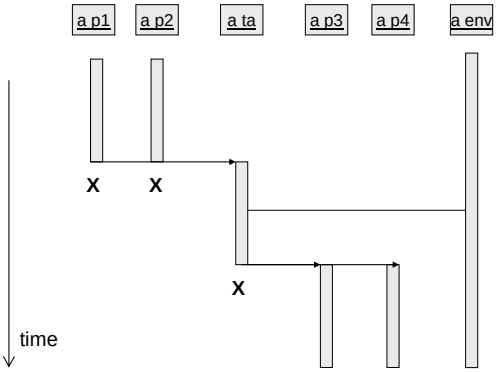


Fig. 7. An example sequence diagram. The input tokens a_p1 and a_p2 are consumed by the transition object a_ta (which can therefore have information about their state). The transition object produces tokens a_p3 and a_p4; it can initialise them with processed information from the inputs, and from the environment a_env. The transition object exists only for the duration of the interaction. X indicates object termination.

3.4 Introducing the environment

Cell behaviour (maturation, differentiation, and division) is affected by environmental factors as well as internal properties. It is important to capture environmental effects in the overall model; that is the purpose of the Environment class in figure 5.

The cell model has multiple levels of detail and abstraction, and the environment can interact with all these. Environmental effects may be global (for example, a global irradiation of tissue), or local (for example, specific to a particular spatial region or type of cell). The environment can affect rates (transition rates within a state model, transition rates within the Petri net model) and behaviours (choice of transition within a state model due to guard values). The environment can itself be affected by the cells (for example, cells excreting chemical signals that are transported by the environment to other cells).

As much, or as little, should be put into the environment as is needed for the particular modelling purpose. This enables environmental effects to be modelled explicitly and rigorously within the same framework as the cell modelling, but cleanly separated from the individual cell models.

4 Combining the layers

The high level Petri net model and the detailed object-oriented models represent the layers of the system. These layers are combined by linking the Petri net place and transition names to the component level class names and state diagram labels. The token classes are named after the Petri net place names, the place classes are named Place_X, where X is the relevant token class name. The start and end points on the state diagram are labelled with the relevant Petri net transition names. The transition classes are named after the Petri net transition name.

This approach provides a simple and straightforward, but nevertheless explicit and rigorous, linking of the two layers of the model. The cell state behaviour, cell differentiation and division, and environmental effects are all modelled. These models make sense individually, at suitable levels of abstraction, and provide a modular approach to modelling and to model validation. How the models are related is clearly defined through this linkage process, and so there is a clear mapping from the model to the code, further easing validation.

Systems biology aims to analyse biological phenomena as dynamic, interacting systems rather than individual components [20, 24]. This high-level approach allows us to address the complexity inherent to biological systems in a way that is impossible using conventional, reductionist molecular biological approaches. A major limitation to the utility of systems approaches to biology is that very many components across multiple scales need to be simultaneously modelled. Building single models that capture multiple levels of biological complexity is extremely difficult. The modelling strategy proposed here provides an intuitive framework for combining models at different biological scales (in this case a tissue differentiation level and a cellular level) in such a way that each level can be treated separately. This separation of levels allows for the construction of manageable models, whilst providing the ability to build multi-scale models.

5 A Domain Specific Language

Our use of different models and notations effectively amounts to a domain specific language (DSL) [10, 29] for modelling cell differentiation and division. As with any DSL, we develop the DSL by extending and adapting existing languages. There are three principle extensions and adaptations.

For the detail of cell behaviours, the models are based on object-oriented modelling notations. However, we apply an agent semantics,

rather than the object semantics defined in, for example, UML2.x [32]. The agent semantics means that an object is instantiated to an active state and allocated its own execution thread. The object persists until it is consumed, and its thread terminates. This alternative semantics allows us to map cleanly from the low-level specification to an agent-based implementation, aiding software engineering validation. Furthermore, this semantics allows us to express concepts in a biologically meaningful way, supporting biological validation.

For the system level, we use Petri nets. However, we modify the semantics of transition firing. In standard Petri nets, a transition fires when a token is in a place. In our usage, the firing of a transition is determined by the state diagram of the relevant place: a token (representing a cell) remains in a place until the conditions are achieved for firing a transition. This may depend on the environment, or simply on passage of time to maturity. Again, this semantic adaptation allows a biologically-meaningful representation of cell behaviours. Note that a common alternative approach is to introduce a timing distribution to control transitions; even if based on biological observation, this is a coarser solution, providing a less biologically-realistic determination of transition rates.

The third key element of our DSL approach is to define the linking between layers (§4). Like UML syntax, concepts in our models are named, and the names are used to link the detailed and system level models, as described above.

The DSL approach opens several novel directions for our research (see also §8). Firstly, we can exploit the metamodelling approach to DSL syntax and semantics, to produce tool support for our models. Secondly, we can exploit the formal underpinnings of Petri nets, and the fact that state diagrams and other OO modelling notations can be represented as Petri nets [30, 45], to support formal analysis of the cell behaviours. The potential of the DSL approach differentiates our simulation and modelling from other multi-model solutions (for example, the work of Harel et al discussed in §1.1), where model integration is done at the implementation level, through a computational framework.

6 Domain model of prostate cell division

6.1 Biological domain

Cancer is a phenotype arising as the result of aberrant interactions between many individual cells. The study of cancer is therefore the study of cell population dynamics. Prostate cancer is the most commonly diagnosed cancer in UK men, accounting for almost 25% of all male cancers in

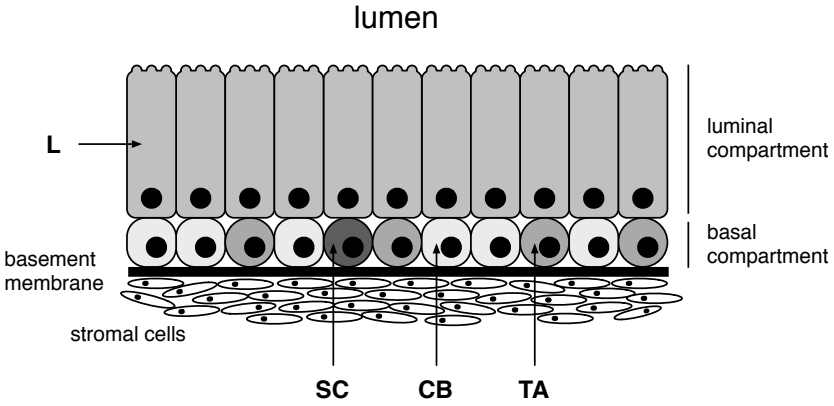


Fig. 8. The major cell types present in the prostate. The stromal cell compartment contains various structures, for example blood vessels, and is not modelled in our work. **SC**: stem cells; **TA**: transit amplifying cells; **CB**: committed basal cells; **L**: luminal cells. The transit amplifying cells in our model include both non-stem basal cells and transit amplifying cells. Figure adapted from [26].

the UK [46]. The UK lifetime risk for getting prostate cancer is approximately 1 in 10 men [46]. Mutations in a wide range of genes (*oncogenes*) are known to increase the risk of cancer [4]. Although the presence of mutations in oncogenes confers an elevated risk of cancer, there is a high level of variability in the timing and exact genotype of cancers. This stochastic element makes the analysis of cancers at the expression level very difficult. Stochasticity and genetic variability make cell population modelling a very attractive tool for the study of cancer neogenesis.

Several cell populations are present in the prostate (figure 8). The majority of prostatic tissue is composed of stromal cells which consist of connective tissues and blood vessels. The tissue of interest with respect to prostate cancer is the prostatic epithelium, which consists of basal, secretory and neuroendocrine cells [6]. The secretory cell population consists of terminally-differentiated columnar cells. The basal cell compartment contains less-differentiated cells that are still in contact with the basement membrane.

The presence of small numbers of self-renewing stem cells in most tissues is now widely accepted. The stem cell population is able to replace dead cells in the tissue by the processes of division and differentiation. Stem cells are able to undergo two types of division: symmetric division in which a single stem cell divides to yield two similar daughter stem

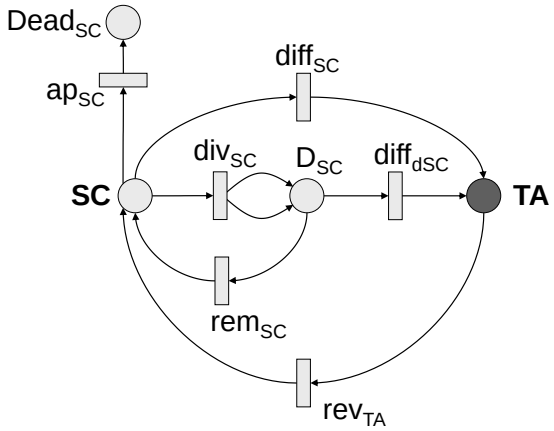


Fig. 9. A Petri net model of stem cell (SC) division and differentiation. A stem cell can divide (div_{SC}); its daughters D_{SC} can each either remain a stem cell (rem_{SC}), or differentiate (diff_{dSC}) to a Transit Amplifying cell TA. A stem cell can also differentiate (diff_{SC}) directly to a TA cell without division, or can undergo apoptosis (ap_{SC}), and become dead (Dead_{SC}). A TA cell can revert (rev_{TA}) to a stem cell.

cells; and asymmetric division in which a single stem cell divides to give rise to a daughter stem cell and a daughter cell of a more differentiated phenotype.

The role of stem cells in the formation and maintenance of tumours is not well understood. Viable mutations in a stem cell will be passed on to its large number of progeny by the normal processes of cell division and differentiation. The cancer stem cell model [25] suggests that, if these stem cell mutations confer a malignant phenotype, then these progeny will form a cancerous population (a tumour). If this model is correct, then the stem cell population is of utmost importance in cancer treatment; the common therapy of removal of the bulk tumour mass will not impact the long-term patient survival rate, whereas ablation of cancerous stem cells would allow successful treatment.

Here we are studying a population of cells moving through a differentiation topology (modelled by the high-level Petri net). The decision-making processes and the underlying molecular biology are encapsulated in the lower level models. The two-level modelling allows us to easily create appropriate agent-based simulations of cell population dynamics.

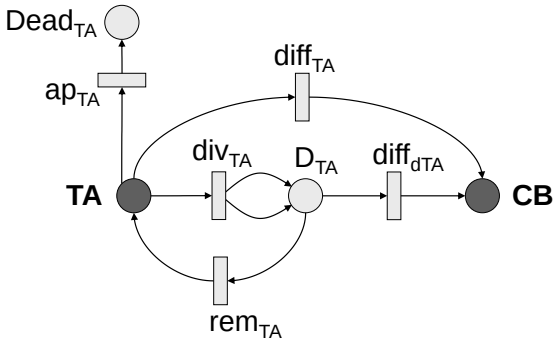


Fig. 10. A Petri net model of transit amplifying (TA) cell division and differentiation. The pattern is similar to the SC diagram, except that a committed basal cell (CB) cannot revert to a TA cell.

6.2 The division pathway model

Figure 9 shows our Petri net model of prostate stem cell division and differentiation. A stem cell can divide; its daughters can each either remain a stem cell, or differentiate to a Transit Amplifying cell TA. In figure 9 the place TA is shown as a *fusion place*, indicating it occurs in another Petri net in this model (here, in figure 10). A stem cell can also differentiate directly to a TA cell without division, or can undergo apoptosis, and become dead. A TA cell can revert to a stem cell.

Figure 10 shows our Petri net model of TA cell division and differentiation. The pattern is similar to the SC diagram, except that a committed basal cell (CB) cannot revert to a TA cell (there is no analogue of the rev_{TA} transition). Note that the dead place is not modelled as a fusion place: each cell type has its own dead place. This is because a dead cell is still present in the tissue, taking up space and potentially influencing different locations of the environment in different ways (for a while at least; a more sophisticated model could have a further transition to a Decayed place).

Figure 11 shows our Petri net model of CB cell division and differentiation. The logic of division and differentiation is identical to the TA cell model in figure 10, but with different cell types and transition types.

Figure 12 shows our Petri net model of luminal cell division. A luminal cell can divide into two luminal cells (although this happens at a very low rate in normal tissue, it is included here because it occurs in the cancerous case). Note we do not need the intermediate daughter cell places in this model, as biologically there is no "remain or differentiate"

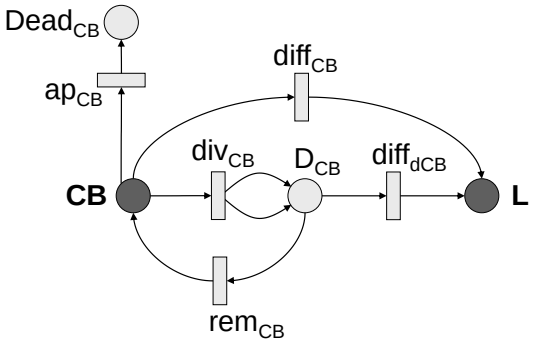


Fig. 11. A Petri net model of committed basal (CB) cell division and differentiation. The logic is identical to the TA cell model in figure 10.

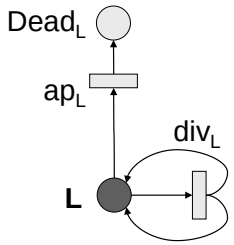


Fig. 12. A Petri net model of luminal (L) cell division. A luminal cell can divide (div_L) into two luminal cells, or can also undergo apoptosis (ap_L), and and become dead (Dead_L).

choice in this division process. A luminal cell can also undergo apoptosis, and and become dead.

6.3 The cell state model

Figure 13 shows the class model of the place tokens, with a subclass for each place type (the daughter cell subclasses and dead cell places are omitted for clarity). This is an instantiation of figure 5, and as such could mostly be automatically derived by a tool. We have chosen to bundle together the similar D classes and the Dead classes into abstract classes, and not show the individual subclasess, for brevity. The only addition compared to figure 5 is the instance variable genome in the abstract Cell class, and the instance variable next in the abstract D class.

The next instance variable indicates whether the daughter cell should preferentially remain as its originator type, or differentiate into the next

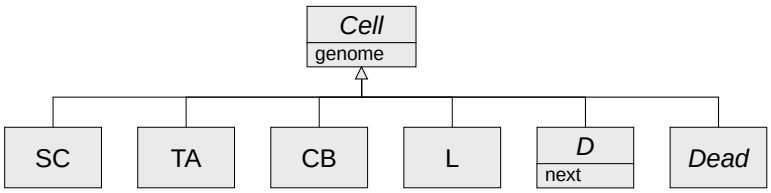


Fig. 13. The class model of the cell tokens (the tokens in the various places). Apart from the Cell and D instance variables, this can be automatically generated from the Petri net model. There is a parallel class model for the places themselves.

type of cell. This captures the asymmetry of the cell division, and leaves enough flexibility for the actual transition taken to be influenced by environmental factors in addition to the value of this instance variable.

The genome instance variable encodes an individual cell's transition rates and mutation rates. The genome is the part of the state that is preserved (possibly slightly modified) as a cell transitions between places. Since the genome is modified at certain points (particularly on division due to copying errors, but also more slowly at at other times), we can model the fact that mutations change an individual cell's transitions rates, which potentially initiate cancerous behaviours. Thus our model does not have a 'flag' that says whether the cell is cancerous or not (cancer is a phenotypical property of a collection of cells, not a property of a single cell), but is a model of a population of cells with varying properties, which allows investigation of the situations that lead to the *emergence* of a cancerous phenotype.

Figure 14 shows our state model of a stem cell SC. It has two sub-states: quiescent and active. It is produced (via reversion, or from a divided daughter cell) in the active state, and it exits to differentiate or divide from the active state. It can apoptose from any state.

Figure 15 shows our state model of a TA cell. It has three sub-states: juvenile (just produced), excreting (active)[1], and mature (ready to divide or differentiate). A TA cell is produced (via differentiation of a stem cell, or from a divided daughter cell) in the juvenile state. It can revert to a stem cell from this juvenile state; it can differentiate or divide from the mature state. It can apoptose from any state.

We omit discussion here of the CB and Luminal L cell state models, for brevity. The state models of the various daughter cells are essentially

[1] The term *excreting* is used here to refer to a metabolically active cell before it commits to differentiation, division or death. Excreting cells in this model can emit signals to the environment.

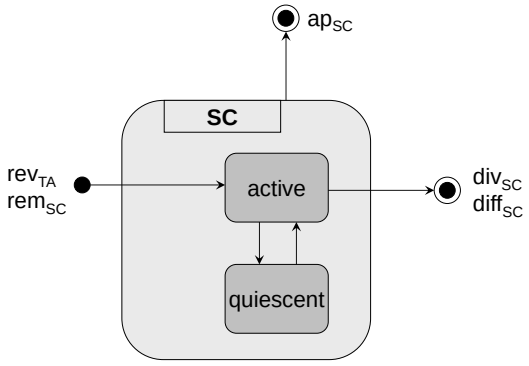


Fig. 14. A state model of a stem cell (a token in the SC place). It has two sub-states: quiescent and active. It is produced in the active state, and it exits to differentiate or divide from the active state. It can apoptose from any state.

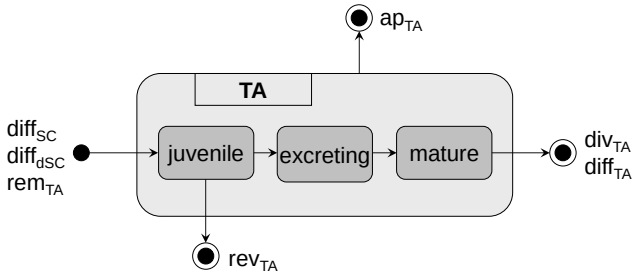


Fig. 15. A state model of a TA cell (a token in the TA place). It has three sub-states: juvenile, excreting, and mature. A TA cell is produced in the juvenile state. It can revert to a stem cell from this juvenile state; it can differentiate or divide from the mature state. It can apoptose from any state.

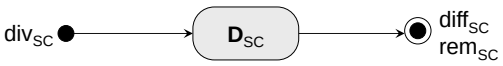


Fig. 16. A state model of a daughter cell (a token in the D_{SC} place). The model for the other daughter cell places has identical logic.

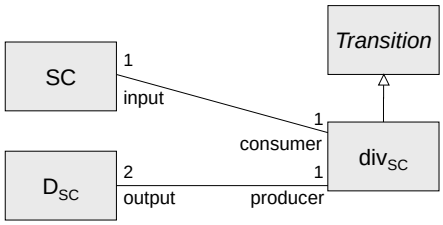


Fig. 17. A class model of the div_{SC} transition. This can be automatically generated from the Petri net model.

trivial (figure 16), having no substates. The state models of the various dead cells are even more trivial (no substates, one input, no outputs).

All the transitions in these state models have conditions and rates associated with them, some affected by environmental conditions. We omit discussion of these, again for brevity.

6.4 The cell transition model

There is a subclass of Transition for each transition in the Petri net (not shown). The subclassing is an instantiation of figure 5, and automatically generatable from the Petri net model. Figure 17 shows our class model of a div_{SC} transition, an (automatically generatable) instantiation of figure 5.

The operation of the specific transition object is to take the genome of the parent stem cell, and 'replicate' it, subject to an environmentally and genomically specified mutation. So, in pseudocode:

```
div_SC()
    output1.genome = mutate(input.genome, env)
    output2.genome = mutate(input.genome, env)
```

This gives two potentially different resulting genomes, as the `mutate()` operation is stochastic.

6.5 The full combined model

All the separate models are combined automatically by matching cell layer and Petri net layer names, as described in §4.

6.6 Model and simulator validation process

Validation is a continuous review process between the biological domain experts, the modellers, and the simulation developers [33]. The valida-

tion process for the models presented in this paper comprised walk-throughs of the diagrammatic models, with highlighting and discussion of the appropriateness of assumptions and abstractions. Meeting notes are recorded in the project wiki.

In the prostate cancer modelling, our domain expert is a group of researchers from Maitland's Yorkshire Cancer Research lab at the University of York. One of the development team is also a member of this lab; 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. In producing the first prototype simulator, there were four major review meetings, at which diagrammatic models were discussed in detail, and changed to better represent the biological understanding of the laboratory researchers. The developer team comprised an implementation expert, two modelling experts and the link-biologist, who has experience of modelling and simulating biological systems. Many of the advances in modelling the prostate cell behaviours occurred in regular discussions among the team.

A validation argument for the models and simulator has been captured in a rigorous form after the modelling [36]. The information needed for this argument was captured during the development process.

7 The platform model and prototype simulator

In moving from the domain model to the platform model, we focus on the implementation detail of the simulator. This requires design decisions about the implementation of the concepts in the domain model.

At the time of writing, we have completed a first prototype simulator, using a platform model developed from the domain model in §6, which was fully validated with YCR biologists before implementation started. The prototype simulator is being calibrated, and we are testing its scalability properties and performance.

7.1 The platform model derived from the domain model

To implement the prostate cell behaviours, we chose to use a process-oriented programming language, since this provides a natural implementation for the state-transition structure of the high-level model in §6. For prototyping, we use the JCSP (Java Communicating Sequential Processes) class library for Java: this also gives us a seamless development from the low-level OO models.

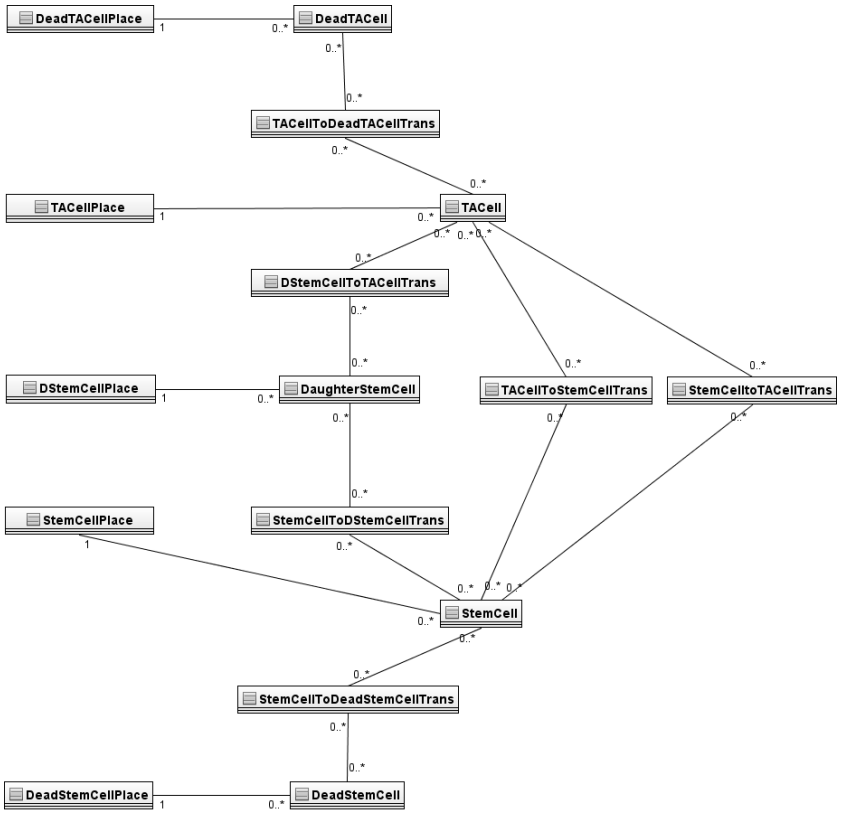


Fig. 18. Platform model of places, transitions, and cells.

JCSP[2] provides a natural way to capture the agent semantics. Any class that implements the JCSP class CSProcess has instances that each run in its own separate Java thread. The Petri net domain model places and transitions map to platform model CSProcess classes, and the Petri net arcs map to communication channels between the processes.

The use of JCSP requires the addition of structures to implement process synchronisation: JCSP barriers to synchronise operations, so that all parts of the simulation experience the same number of time-steps, at the same time.

The prototype simulation is developed from the stem cell division and differentiation domain model (figure 9), with the addition of transit amplifying cell apoptosis (figure 10). The platform model is summarised

[2] http://www.cs.kent.ac.uk/projects/of a/jcsp/

in a Java class diagram (figure 18). Each place and transition from the domain model is a singleton Java class that implements the JCSP class CSProcess. Because places and transitions have common characteristics, inheritance hierarchies are used (not shown in figure 18).

The current prototype simulator includes stem cells (SCs), daughter stem cells, TA cells, and dead stem and TA cells. Stem cells move between the substates active and quiescent, and can apoptose, differentiate or divide (figure 14). The TA cells have only limited behaviour: they are either active or resting, and can only revert to a stem cell or apoptose.

Cell division and differentiation in the platform model is derived from the Petri net structures, with the detailed actions of each behaviour defined in the state diagrams of the domain model. The sequence diagrams in the domain model (not illustrated) provide the interaction detail for the implementation.

The cell state and operations are defined on the hierarchy of Cell classes (implementing figure 13) Each cell type has appropriate methods for changing its state, and an instance of the Genome class. The design decision to extract the genome as a Java class supports the operations needed to enact the state-diagram state changes (such as the move from active to quiescent states of the stem cell, figure 14). The genome stores and sets probabilities for different cell events, which gives the necessary control over state change for biological experimentation with the simulator; as in the cell biology, the genome representation allows differentiation to change the 'active genes' in the genome.

The platform model also has a basic Environment class. This is used to represent a global environment, the simulated prostate tissue. Cells can poll the environment at any time, and the results affect the specific behaviour of each cell. The interaction of the cells with the environment will be used to simulate global mutation rates, and ultimately potential treatments that could alter the mutation rate of the whole prostate tissue. There is significant further development to be done, of which the refinement of the environment, and of the behaviour and regulation of the genome during the life of a cell are perhaps the most interesting and potentially challenging.

7.2 The platform model implementation additions

The platform model needs extra implementation specific features not derived from the domain model: buffers between places and transitions; JCSP barriers to synchronise processes; a global clock to simulate time.

JCSP multi-threading provides better performance than a purely-sequential implementation. Threads are used for places and transitions, whilst individual cells (tokens in each location) are implemented as Java

objects (not CSProcess objects, due to the limited number of threads available), with relevant behaviours invoked each simulation timestep. This means that the cells in a place or transition can all be processed during each time-step. This design gives the potential to support many thousands of cell objects per place, which is the biological requirement for the eventual cancer simulations.

The Place and Transition JCSP CSProcess classes provide access to both the read and write CSP channels (and barriers) as well as interaction with the GlobalClock and the Environment singleton classes. Specific places and transitions in the platform model extend the Places and Transition classes: for example, TACellPlace extends Place implements the domain model place TA in figure 9, and TACellToStemCellTrans extends Transition implements revTA in figure 9.

In the implementation, cells are passed between places and transitions via JSCP InfiniteBuffers, which are used to control synchronisation of channels connecting places and transitions: this means that there is one InfiniteBuffer for each arc in figure 9. InfiniteBuffers do not impose read or write locks on the threads accessing them (unlike other JSCP channels); this means that threads can write data into the buffer and proceed without waiting for another process to read the data. In the simulator, this is of particular importance when there are no cells arriving at a particular location in a particular time-step: the place can process any cells that are already there, rather than be suspended waiting for cells that are not coming.

To keep track of the number of timesteps that the simulation has completed, the GlobalClock is implemented as a JCSP CSProcess. The clocking allows state readings to be regularly output during simulation, and allows the setting of an end point for the simulation run. The GlobalClock synchronises on the same barrier as all the places and transitions.

The barrier design and implementation is critical in ensuring that all places and transitions experience the same sequence of timesteps and remain synchronised. Before any place or transition can accept cells from a buffer to which it is connected, all places and all transitions must be synchronised on a read barrier. Once synchronised all cells are removed from the buffers and are processed for that timestep. Once the cells are processed all places and all transitions that output cells (that is, all locations except DeadPlaces) must synchronise on a write barrier. Once synchronised, the locations move the relevant cells to the correct buffer, and then synchronise on the read barrier. This ensures that all cells are read before any are processed, and all cells are written correctly before any are read. The additional development burden of correct barrier design can be verified by formal CSP analysis if required.

All the places and transitions are modelled as singleton classes so that they can access each other as the program requires, supporting essential feedback mechanisms in the simulation. For example, in the prototype model, the probability of a TA cell reverting to a stem cell increases if there are no stem cells in the stem cells place, and stem cells remain quiescent longer as the number of TA cells rises; these rate changes act as a surrogate for spatial crowding in the environment (see [36] for further discussion).

7.3 Initial Results

The prototype simulator has been used to develop an approach to implementation from multiple linked models. The simulator is subject to testing, particularly of the logic of the design and performance. It has not yet been used to run biologically-relevant experiments.

The implemented genome was designed to calibrate the prototype simulator, to make it produce large numbers of TA cells and maintain relatively small numbers of stem cells, with low death rates. This will replicate the normal behaviour of the prostate in biological experimentation.

The prototype simulator has completed testing with total cell numbers ranging between 100 000 to 600 000. Typical performance is illustrated by initialising a simulation run with 2 stem cells; the number of cells quickly grows as stem cells divide and differentiate; the expansion then slows as the TA population gets large, due to the surrogate crowding effects; the result after 100 000 time-steps is 851 stem cells, 598 616 TA cells, 56 dead stem cells, and 28 153 dead TA cells. (These numbers are not being represented as biologically relevant, since the rates have not been calibrated, but they do give an indication of the achievable scale.) This simulation took 23 minutes on a 3.2GHz AMD Phenom II computer. Whilst this is not particularly fast, it is a significant improvement on the time taken to conduct the equivalent laboratory tests on prostate cell behaviour (weeks or months of expensive wet-lab procedures).

Simulation performance is likely to decrease as simulation complexity increases but it should be possible to simulate populations of cells well into the millions within realistic timeframes. Further work on the implementation architecture to capture more of the natural parallel structure of the model (for example, by increasing the implementation parallelism within individual places and transitions) would support distribution across clusters or cloud platforms. The full process-oriented design, with each individual cell implemented as a process, would be transferable to high-performance parallel languages such as occam-π [50] not limited by the number of available threads.

8 Further work

8.1 Further work on the prostate model

The prostate is a complex organ composed of many cell types. The model we outline in this work is an excellent foundation for an iterative systems biology programme to analyse prostate cancer neogenesis.

Parameterisation & Initial Conditions. As with any model, we have a large number of possible variables that need to be parameterised. Transition rates in the Petri net layer need to be defined as accurately as possible. Similarly, the initial conditions for the model need to be sensibly defined. These parameter values come from a variety of sources: literature searches; close collaboration with domain experts when choosing suitable *in silico* proxies; focussed wet-lab experimentation to generate the required data.

Cell Genome. Currently, each cell type has a minimal genome. As the model is refined, suitable variables can be placed upon the genome, allowing for mutable cell phenotypes. It is tempting to add as much complexity to the model as possible, thus initially adding all the genes that we can think of to the cell's genome; however, we should attempt to keep the model as simple as possible whilst retaining the ability to address real biological hypotheses. The addition of genes to the cell genome and the parameterisation of their rates is a major challenge for the next stage of this project.

New Cell types. Castration-resistant prostate cancer frequently shows a neuroendocrine like phenotype [47]. Currently, our model does not include stromal or neuroendocrine cells, with the interactions between these cell types and the modelled cells being abstracted to 'global' signals. If necessary, the inclusion of stromal and neuroendocrine cells would be possible, but this extra complexity would have to be justified by added ability of the model to address specific biological hypotheses.

8.2 Tool development

In this paper, the way the models in the two layers are combined is described informally. The approach could readily be made rigorous using model-weaving techniques from model driven engineering [2, 3, 21], allowing tool support (see also §5).

The (draft) Petri net ISO standard [17, 18] defines the notation in terms of a UML class diagram, or metamodel. This would make it possible to integrate, at a deeper semantic (or at least abstract syntax) level, the Petri net notation and UML state diagrams (as well as other UML modelling notations). Two motivations for a language integration would be semantic consistency, and integrated tool development. Support tools would obviously help the use of combinations of models. Semantic consistency is theoretically desirable; however, even if we unify the semantics of two diagram notations, there is no way of guaranteeing that the user intends their diagram to follow the semantics defined by the language developer.

8.3 Extensions to this modelling approach

There are several extensions to this Petri net plus object modelling approach that are needed to make this more generally applicable to biological modelling. We outline a few here.

Petri nets can also be used to model biological processes of inhibition and catalysis, by suitably interpreting the relevant arcs. Inhibitory arcs need no change to the model described: they have the Petri net semantics of not permitting a transition to fire if there is a token in the relevant place. Catalytic arcs[3] need one change to the sequence diagram: the catalytic token object is not terminated (not consumed) by the transition. For example, see figure 19.

Larger models will need more structuring capabilities. We have used fusion places (figure 2) to split our Petri net into digestible chunks. Fusion transitions could be used to capture an entire sub-net, for example, to capture the identical logic of the various cell division stages. This requires further work to determine naming conventions to allow the individual models to be combined into a well-defined whole.

8.4 Incorporating other approaches

There is nothing specific to Petri nets and object models in our general approach; they are simply one way of capturing some high level global structure, and lower level component behaviours. This choice is a natural fit to a cell division model, but other biological processes may be better fit by other modelling techniques.

[3] Catalysis is sometimes denoted by a double-headed solid arc in the Petri net, rather than the dashed arc of figure 19a. In our approach, this would imply that the token is consumed, and a new different token produced to replace it. The double-headed arc fits more readily into the 'token flow' interpretation (see §2.1), rather than the 'production and consumption' interpretation.

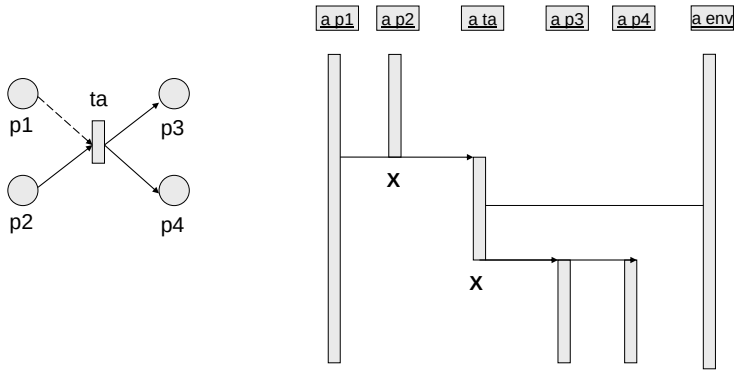


Fig. 19. An example catalytic arc. This is a small variant of figure 1, with the arc from `p1` to `ta` here a catalytic arc (left). The only change to the overall model is in the sequence diagram (right), where the token `p1` is not consumed by the transition.

For example, L-Systems [37] have been designed to model plant-like growth process. Preliminary work suggests that they can be used in a hybrid model in an analogous way.

At the highest level, an L-System component (called a *module*) is just a symbol; and *productions* (rewrite rules) define how symbols 'grow' (are rewritten into sequences of symbols).

Descending levels one can add detail. Symbols can have parameters whose values are determined by the productions. As one adds more detail, full blown object/simulation models can be added to give the symbols and rewrites richer behaviours. This is the approach that the L-Studio tool [23] implements (by calling out to C functions).

We can use our approach to make this interaction of rewrite rules and object models fully rigorous: a high level L-System growth model can be combined with a lower level object-based module model. The correspondence between the Petri net approach and the L-System approach is shown in figure 20 (and hints at the possibility of a unifying metamodel). We can use the same approach to develop an object model that describes complicated behaviour in components and in component transitions. The 'glue' between the L-System and object model layers would need to be a little different from our Petri net/object model glue described above, however, as it would need to incorporate communication between the low level objects (between symbol instances), eg to implement hormone transport, during the non-growth part of the iteration.

concept	Petri net	L-System
type of component	place	symbol
change of component	transition	production
instance of component	token	instance of symbol
current state	marking	current generation

Fig. 20. Relationship between high level Petri net models and L-System models

9 Summary and Conclusions

We have presented a rigorous hybrid modelling approach combining the strengths of Petri nets and object-oriented models. These models make sense individually, at suitable levels of abstraction, and provide a systematic modular approach to modelling and to model validation, allowing us to use diverse modelling techniques in developing multi-scale models with no loss of rigour or validity. This hybrid modelling approach is not specific to Petri nets and object modelling; we outline a similar approach to combining L-Systems models with other modelling approaches, and discuss how the hybrid approach effectively amounts to the use of a domain specific language.

We have developed our hybrid modelling approach in order to model cell division and differentiation in the prostate. In building the prostate cell model, we have followed the CoSMoS process, with close cooperation between the biological domain experts and the modellers and simulation developers, to ensure that the model makes both biological and computational sense. In a companion paper [36] we have argued the validity of our model. We have developed a prototype simulation of the prostate cell model, and demonstrated that it is capable of simulating biologically-relevant numbers of cells.

In future work we will develop the hybrid modelling approach as a generic technique for building multi-scale models, along with tool support. We will also develop the prostate cell model and simulation further, and use it to investigate the onset of cancerous phenotypes.

Acknowledgements

This work is supported by Program Grant support (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 thank the members of the Cancer Research Unit in York for their invaluable input to the project. Thanks to the anonymous referees for helpful suggestions to improve the paper.

References

[1] Paul S. Andrews, Fiona A. C. Polack, Adam T. Sampson, Susan Stepney, and Jon Timmis. The CoSMoS process, version 0.1: A process for the modelling and simulation of complex systems. Technical Report YCS-2010-453, Department of Computer Science, University of York, March 2010.

[2] Jean Bézivin, Nicolas Farcet, Jean-Marc Jézéquel, Benot Langlois, and Damien Pollet. Reflective model driven engineering. In *UML 2003 – The Unified Modeling Language*, volume 2863 of *LNCS*, pages 175–189. Springer, 2003.

[3] Jean Bézivin, Frédéric Jouault, Peter Rosenthal, and Patrick Valduriez. Modeling in the large and modeling in the small. In *Model Driven Architecture*, volume 3599 of *LNCS*, pages 901–901. Springer, 2005.

[4] L. C. Cantley, K. R. Auger, C. Carpenter, B. Duckworth, A. Graziani, R. Kapeller, and S. Soltoff. Oncogenes and signal transduction. *Cell*, 64:281–302, 1991.

[5] Claudine Chaouiya. Petri net modelling of biological networks. *Briefings in Bioinformatics*, 8(4):210–219, 2007.

[6] A. T. Collins and N. J. Maitland. Prostate cancer stem cells. *European Journal of Cancer*, 42:1213–1218, 2006.

[7] Sol Efroni, David Harel, and Irun R. Cohen. Reactive animation: Realistic modeling of complex dynamic systems. *Computer*, 38:38–47, 2005.

[8] Sol Efroni, David Harel, and Irun R. Cohen. A theory for complex systems: reactive animation. In Ray Paton and Laura A. McNamara, editors, *Multidisciplinary Approaches to Theory in Medicine*, volume 3 of *Studies in Multidisciplinarity*, pages 309 – 324. Elsevier, 2005.

[9] Martin Fowler. *UML Distilled: brief guide to the standard object modeling language*. Addison-Wesley, 3rd edition, 2004.

[10] Martin Fowler. *Domain Specific Languages*. Addison-Wesley, 2010.

[11] Philip Garnett, Susan Stepney, Francesca Day, and Ottoline Leyser. Using the CoSMoS process to enhance an executable model of auxin transport canalisation. In Stepney et al. [44], pages 9–32.

[12] Philip Garnett, Susan Stepney, and Ottoline Leyser. Towards an executable model of auxin transport canalisation. In Susan Stepney, Fiona Polack, and Peter Welch, editors, *Proceedings of the 2008 Workshop on Complex Systems Modelling and Simulation*, pages 63–91. Luniver Press, 2008.

[13] P. P. Glory, N. G. David, and J. D. Emerald. Petri net models and non linear genetic diseases. In *Bio-Inspired Computing: Theories and Applications (BIC-TA), 2010*, pages 1466–1470. IEEE, 2010.

[14] David Harel. Statecharts: a visual formalism for complex systems. *Science of Computer Programming*, 8:231–274, 1987.

[15] David Harel, Sol Efroni, and Irun R. Cohen. Reactive animation. In *Formal Methods for Components and Objects 2002*, volume 2852 of *LNCS*, pages 136–153. Springer, 2002.

[16] David Harel and Yaki Setty. Generic reactive animation: Realistic modeling of complex natural systems. In *Formal Methods in Systems Biology*, volume 5054 of *LNBI*, pages 1–16. Springer, 2008.

[17] High-level Petri Nets - Concepts, Definitions and Graphical Notation. International Standard ISO/IEC 15909, 2000. Final Committee Draft: www.petrinets.info/docs/pnstd-4.7.4.pdf.

[18] Software and Systems Engineering High-level Petri Nets Part 2: Transfer Format. International Standard ISO/IEC 15909, 2005. WD 15909-2:2005(E): www.petrinets.info/docs/ISO-IEC15909-2.WD.V0.9.0.pdf.

[19] Kurt Jensen and Lars M. Kristensen. *Coloured Petri Nets*. Springer, 2009.

[20] H. Kitano. Systems biology: a brief overview. *Science*, 295(5560):1662–1664, 2002.

[21] Dimitrios Kolovos, Richard Paige, and Fiona Polack. Merging models with the Epsilon Merging Language (EML). In *Model Driven Engineering Languages and Systems*, volume 4199 of *LNCS*, pages 215–229. Springer, 2006.

[22] Hillel Kugler, Antti Larjo, and David Harel. Biocharts: a visual formalism for complex biological systems. *Journal of The Royal Society Interface*, 7(48):1015–1024, 2010.

[23] L-Studio. Plant modeling with CPFG virtual laboratory. algorithmicbotany.org/lstudio/flyer.pdf.

[24] Y. Lazebnik. Can a biologist fix a radio? — or, what I learned while studying apoptosis. *Cancer Cell*, 2:179–182, 2002.

[25] N. J. Maitland and A. T. Collins. A tumour stem cell hypothesis for the origins of prostate cancer. *BJU International*, 96(9):1219–1223, 2005.

[26] N. J. Maitland and A. T. Collins. Prostate cancer stem cells: a new target for therapy. *Journal of Clinical Oncology*, 26(17):2862–2870, 2008.

[27] Wayne Materi and David S Wishart. Computational systems biology in cancer: Modeling methods and applications. *Gene Regulation and Systems Biology*, 1:91–110, 2007.

[28] Hiroshi Matsuno, Masao Nagasaki, and Satoru Miyano. Hybrid Petri net based modeling for biological pathway simulation. *Natural Computing*, pages 1–22, 2009. doi:10.1007/s11047-009-9164-6.

[29] M. Mernik, J. Heering, and A.M. Sloane. When and how to develop domain-specific languages. *ACM Computing Surveys*, 37(4):316–344, 2005.

[30] H. Motameni, A. Movaghar, M. Siasifar, M. Zandakbari, and H. Montazeri. Mapping state diagram to Petri net : An approach to use Markov theory for analyzing non-functional parameters. In *Advances and Innovations in Systems, Computing Sciences and Software Engineering*, pages 185–190. Springer, 2007.

[31] M Nagasaki, A Doi, H Matsuno, and S Miyano. Petri net based description and modeling of biological pathways. In *Algebraic Biology 2005*, pages 19–31. Universal Academy Press, 2005.

[32] OMG. Unified Modeling Language (UML). www.omg.org/spec/UML/.

[33] Fiona A. C. Polack. Arguing validation of simulations in science. In Stepney et al. [44], pages 51–74.

[34] Fiona A. C. Polack, Paul S. Andrews, Teodor Ghetiu, Mark Read, Susan Stepney, Jon Timmis, and Adam T. Sampson. Reflections on the simulation of complex systems for science. In *ICECCS 2010*, pages 276–285. IEEE Press, 2010.

[35] Fiona A. C. Polack, Paul S. Andrews, and Adam T. Sampson. The engineering of concurrent simulations of complex systems. In *CEC 2009*, pages 217–224. IEEE Press, 2009.

[36] Fiona A. C. Polack, Alastair Droop, Philip Garnett, Teodor Ghetiu, and Susan Stepney. Simulation validation: exploring the suitability of a simulation of cell division and differentiation in the prostate. In Susan Stepney, Peter Welch, Paul S. Andrews, and Carl G. Ritson, editors, *Proceedings of the 2011 Workshop on Complex Systems Modelling and Simulation, Paris, France, August 2011*. Luniver Press, 2011.

[37] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The Algorithmic Beauty of Plants*. Springer, 1990.

[38] Mark Read, Paul S. Andrews, Jon Timmis, and Vipin Kumar. A domain model of experimental autoimmune encephalomyelitis. In Susan Stepney, Peter H. Welch, Paul S. Andrews, and Jon Timmis, editors, *Proceedings of the 2009 Workshop on Complex Systems Modelling and Simulation, York, UK, August 2009*, pages 9–44. Luniver Press, 2009.

[39] Mark Read, Paul S. Andrews, Jon Timmis, and Vipin Kumar. Using UML to model EAE and its regulatory network. In *8th International Conference on Artificial Immune Systems (ICARIS)*, volume 5666 of *LNCS*, pages 4–6. Springer, 2009.

[40] Magali Roux-Rouquié, Nicolas Caritey, Laurent Gaubert, and Camile Rosenthal-Sabroux. Using the Unified Modelling Language (UML) to guide the systemic description of biological processes and systems. *BioSystems*, 75:3–14, 2005.

[41] Derek Ruths, Melissa Muller, Jen-Te Tseng, Luay Nakhleh, and Prahlad T. Ram. The signaling Petri net-based simulator: A non-parametric strategy for characterizing the dynamics of cell-specific signaling networks. *PLoS Comput Biol*, 4(2):e1000005, 2008.

[42] Yaki Setty, Irun R. Cohen, Avi E. Mayo, and David Harel. On using divide and conquer in modeling natural systems. In Anne Condon et al., editors, *Algorithmic Bioprocesses*, pages 661–674. Springer, 2009.

[43] Yong-Jun Shin and Mehrdad Nourani. Statecharts for gene network modeling. *PLoS ONE*, 5(2):e9376, 2010.

[44] Susan Stepney, Peter Welch, Paul S. Andrews, and Adam T. Sampson, editors. *Proceedings of the 2010 Workshop on Complex Systems Modelling and Simulation, Odense, Denmark, August 2010*. Luniver Press, 2010.

[45] Ivana Tričković. Formalizing activity diagram of UML by Petri nets. *Novi Sad J. Math.*, 30(3):161–171, 2000.

[46] Cancer Research UK. Prostate cancer — UK incidence statistics. info.cancerresearchuk.org/cancerstats/types/prostate/incidence/. accessed 18/01/2011.

[47] N. Vashchenko and P. A. Abrahamsson. Neuroendocrine differentiation in prostate cancer: implications for new treatment modalities. *European Urology*, 47:147–155, 2005.

[48] Ken Webb and Tony White. Combining analysis and synthesis in a model of a biological cell. In *Proceedings of the 2004 ACM symposium on Applied computing*, SAC '04, pages 185–190. ACM, 2004.

[49] Ken Webb and Tony White. UML as a cell and biochemistry modeling language. *BioSystems*, 80:283–302, 2005.

[50] Peter H. Welch and Frederick R. M. Barnes. Communicating mobile processes: introducing occam-pi. In *25 Years of CSP*, pages 175–210. Springer, 2005.

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 development 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) 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

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

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 model used in developing the first-phase prototype 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

This section reviews our existing work on validity arguments, and introduces a notation for summarising arguments.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.

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

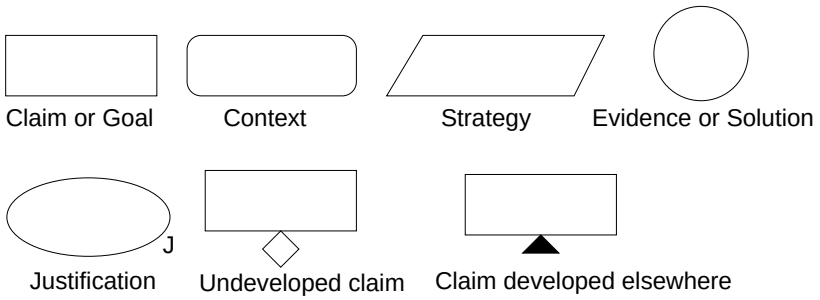


Fig. 1. Basic GSN notations [13, 23]. In safety case arguments, undeveloped *goals* are always subsequently expanded to *solutions*. This is not the case in validity arguments, where claims may be left undeveloped; in validity arguments, end point is a reference to *evidence* to substantiate the claim.

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.

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.

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 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.

This section focuses on 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 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] which presents a generic argument over the results of a simulation.

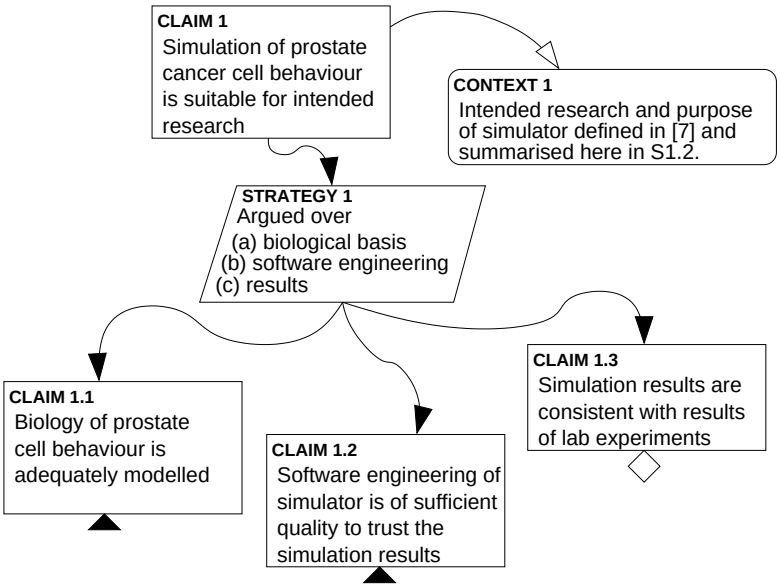


Fig. 2. A top-level argument for the adequacy of the simulation.

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, including artificial daughter cells, added to the domain model during discussion with domain experts of how to model different effects and influences on division and differentiation. 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

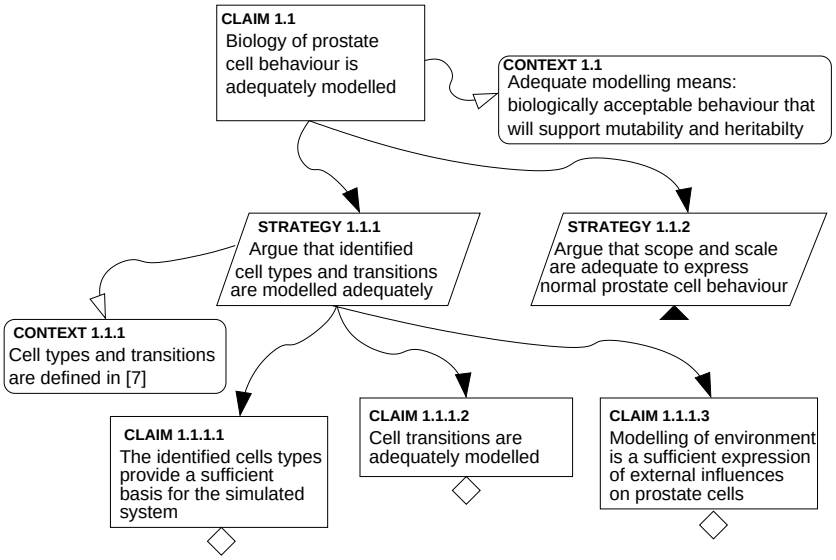


Fig. 3. Expanding Claim 1.1 (Figure 2) to consider adequate modelling of prostate cell behaviours (as described in [7]).

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.

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 developers 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

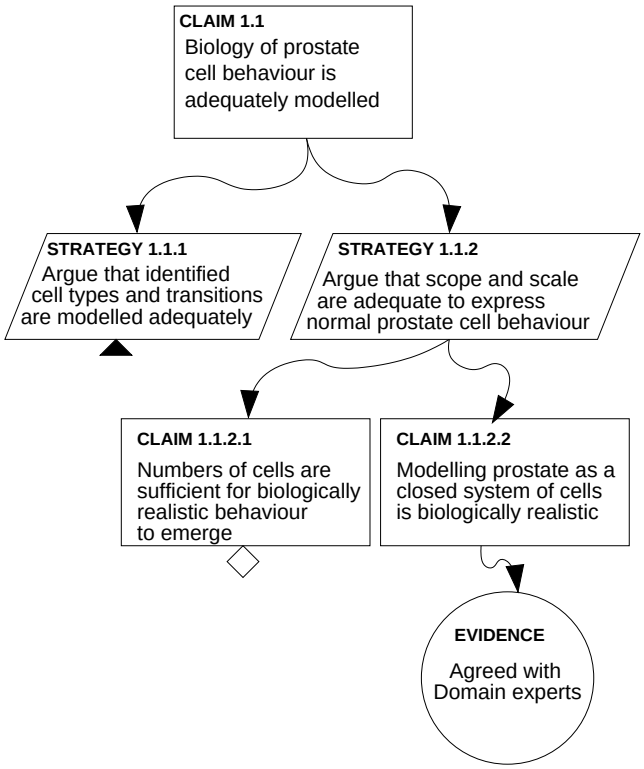


Fig. 4. Expanding Claim 1.1 (Figure 2) to consider adequate scope and scale of prostate cell models.

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.

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 a systematic development from abstract models 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 re-

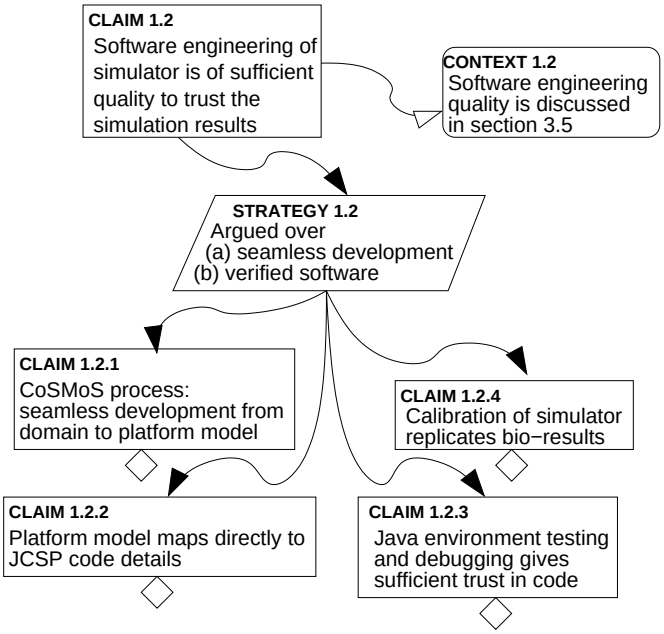


Fig. 5. Expanding Claim 1.2 (Figure 2) concerning Software Quality

viewing how the domain model concepts map to concepts in the platform model. Similarly, Claim 1.2.2 requires demonstration 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 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 simulator 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 simulator 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 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 to 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.

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 would be 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. The 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 or 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. In work using the CoSMoS principled approach and 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 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 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.

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. Most of the modelled cell types map well to biologically distinct (stem and luminal) or distinguishable (committed basal and transit amplifying) cell types, but the domain model also includes 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 of the domain model 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: this is the focus of the first phase of the project, with the ability to populate a naive prostate calibrated against laboratory data on prostate development.

As a last point, the representation of the domain introduces the surrogacy problem. Each concept in the simulator has 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, sensitivity analysis, and 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 does not relate directly to the biological data, as 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 environmental inputs, and needs careful validation through calibration and, possibly, through comparison of spatial and aspatial versions of the simulation.

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 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 (e.g. if someone were ill-advisedly to attempt to use the simulation evidence unsupported by laboratory evidence, to develop a new cancer intervention), 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 could be used to challenge models and identify assumptions [17].

3.5 Simulation engineering

The engineering quality of a 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 explicit the need to validate the implementation, 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 software engineering validation strategies used here 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. 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 cell 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 (e.g. 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 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 model, 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).

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 simulator (described in [7]), showing how an understanding of the fitness for purpose and limitations of a simulator for use 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 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. [8, 9]) 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 safety case argumentation research, including recent work on side arguments and the justification of strategies and claim inter-dependency. 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 and 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 Programme 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.

Multiscale pairwise approximations for ecological modelling

Rebecca Mancy, Simon Rogers, and Patrick Prosser

School of Computing Science, University of Glasgow, G12 8QQ
rebecca.mancy@glasgow.ac.uk

Abstract. Spatial aspects of interaction are important for many complex systems. However, the analytic intractability of spatially explicit simulations makes alternatives that capture key characteristics of these systems attractive. In ecological modelling, pairwise approximations are often used (e.g. evolutionary game theory on graphs); however, with one exception, these limit interactions to nearest neighbours. We develop an alternative to the multiscale approach of [1] and compare these methods with one another and a spatially explicit multi-scale simulation.

Spatial effects are known to influence the macroscopic behaviours of a range of complex systems, including many ecological processes. For example, whilst evolutionary processes invariably select against altruism in non-spatial (mean field) models, they select for altruism under a range of conditions in spatial models e.g. [2]. A natural modelling paradigm is that of spatially explicit stochastic models (e.g. probabilistic cellular automata) but analytic intractability and extended run-times have encouraged the development of approximations that benefit from increased analytic tractability.

1 Pairwise Approximations

Pairwise approximation models have been used to model the behaviours of complex systems in spatial settings and have been especially productive in helping answer a range of questions in theoretical biology where spatial competition between strains is of interest. In this context, the standard approach considers organisms that live on a network of sites, with neighbourhoods defined for the processes of *dispersal* and *interaction*. Individuals can disperse offspring to sites within a neighbourhood, or can be influenced by organisms living there.

The idea underpinning pairwise approximations is to trace the proportion of pair types over time. In a model of competitive interactions

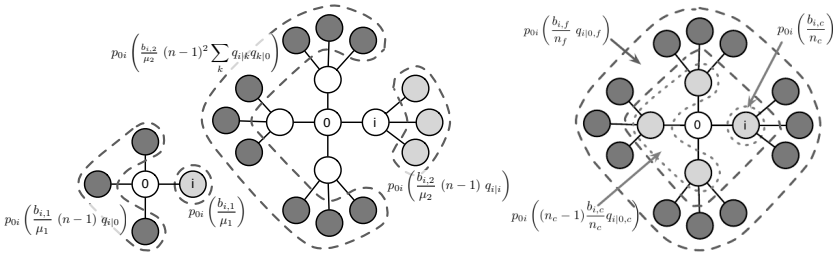
between altruists and egoists, we trace the proportion of *altruist-altruist*, *altruist-egoist* and *egoist-egoist* neighbour pairs. A differential equation with respect to time $\dot{p}_{ij}$ is derived for the proportion of each pair type ij, and the system solved to find the steady state proportion of ij pairs. From this, overall proportions of altruists and egoists can be derived to determine which characteristic is selected.

Given that ecological processes (e.g. seed dispersal) act at a range of scales, pairwise approximations have been critiqued for limiting interactions to immediate neighbours. In [1], a multiscale pairwise approximation technique is described that accounts for two or more spatial scales by defining processes on disjoint neighbourhoods. For example, pair densities at two distances could be denoted by $p_{ij,c}$ (close) and $p_{ij,f}$ (far). This approach requires a rate equation for each of the pair types at each of the two interaction distances, so the number of equations grows linearly with the number of neighbourhoods considered.

2 Alternative Multiscale Approach

We investigate the use of an alternative approach to incorporating a range of spatial scales into a pairwise approximation on a regular lattice or random regular graph, as complementary method to direct simulation. Our approach is based on a direct extension of the standard model and assumes that densities of neighbour types two steps away can be approximated by calculating the expected densities of strains one step away from the *neighbours* of the focal site. In other words, it assumes that two-step densities can be approximated by averaging over one-step densities. It can account for an arbitrary number of interaction distances (number of links) without additional equations and thus more closely approximates a continuous space model. However, in comparison with continuous space models, the use of a network means that the model can represent structures other than Euclidean space, important for modelling structures such as social networks or heterogeneous environments.

For simplicity, we follow [3] and assume that death occurs at a constant rate d_i for each strain i (i.e. death is independent of crowding; c.f. [1]). Each strain i has a total birth rate for each interaction distance δ (rate of offspring production dispersed to that distance), denoted $b_{i,\delta}$, such that birth rate into a *particular* site at distance is given by $b_{i,\delta}/\mu_\delta$, where the number of neighbours at distance δ is given by $\mu_\delta = f(n,\delta)\frac{n!}{(n-\delta)!}$. Note that $f(n,\delta)$ depends on the network topology and is used to scale the number of neighbours to avoid double-counting sites that can be reached via two routes. A full symbol list is shown in Tab. 1.



(a) Our approach (b) Ellner (2001)

Fig. 1. (a) Birth rates of species i into focal empty site (marked 0) of an *0-i* pair, from the one-step and two-step neighbourhoods and $n = 4$. Rates are found by multiplying the birth rate at the appropriate distance $b_{i,\delta}$ by the local density of sites in state i, then by the global density of *i-0* pairs, p_{i0}. For example, a birth into the empty site from an organism on the LHS two steps away (not passing through the i on the RHS of the known pair), is given by the sum of birth rates of triples p_{i10}, p_{i20}, etc. Since we only model pairs and assume independence from non-immediate neighbours, we approximate this as $p_{ik0} \approx q_{i|k} q_{k|0}$, resulting in the sum $\sum_k q_{i|k} q_{k|0}$. (b) Illustration of [1], here shown with our notation and c denoting one-step neighbourhood and f denoting two-step neighbourhood (values selected for illustration only). For f, information about the known i and about the distance from the focal site is not explicitly used.

Figure 1 illustrates rate derivation for one-step and two-step neighbourhoods, and leads to the following differential equation for $\dot{p}_{ii}$ (the full system includes equations for $\dot{p}_{0j}$, $\dot{p}_{00}$ and $\dot{p}_{ij}$ where $i \neq j$; three-step neighbourhoods etc. are derived similarly).

$$\begin{aligned}\tfrac{1}{2}\dot{p}_{ii} = {} & p_{0i}\left(\frac{b_{i,1}}{\mu_1} + \frac{(n-1)b_{i,1}}{\mu_1} q_{i|0}\right) \\ & + p_{0i}\left(\frac{(n-1)b_{i,2}}{\mu_2} q_{i|i} + \frac{(n-1)^2 b_{i,2}}{\mu_2} \sum_k q_{i|k} q_{k|0}\right) - d_i p_{ii}\end{aligned}$$

Before using this approach in applied contexts, it is important to explore any differences between our approach and that of [1], as well as differences in qualitative or quantitative predictions between this approach and spatially explicit stochastic simulations. This poster presents the results of these tests to establish the conditions under which this approximation is appropriate, as well as the effect of different functional forms for $f(n, \delta)$ for different network topologies.

Table 1. Symbols used (nh = neighbourhood)

Symbol	Explanation	Symbol	Explanation
n (n_c)	No. (close) neighbours	μ_δ	No. sites at distance δ
d_i	Death rate strain i	$b_{i,\delta}$ $(b_{i,c})$	Birth rate to dist. δ (close nh)
p_{ij} $(p_{ij,c})$	Global density (close) i-j pairs	$q_{i\|j}$ $(q_{i\|j,c})$	Density of i in (close) nh of a j

Acknowledgements This work was supported by an EPSRC PhD studentship.

References

[1] S. Ellner. Pair Approximation for Lattice Models with Multiple Interaction Scales. *Journal of Theoretical Biology*, 210(4):435–447, June 2001.

[2] Laurent Lehmann and François Rousset. How life history and demography promote or inhibit the evolution of helping behaviours. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 365(1553):2599–2617, September 2010.

[3] M. van Baalen and D. Rand. The Unit of Selection in Viscous Populations and the Evolution of Altruism. *Journal of Theoretical Biology*, 193(4):631–648, August 1998.

Developing an *in silico* tool to explore the mechanisms behind mucosal lymphoid tissue formation

Kieran Alden[1,2], Paul S. Andrews[2], Jon Timmis[2,3], Henrique Veiga-Fernandes[4], and Mark Coles[1]

[1] Centre for Immunology and Infection, Hull York Medical School and Department of Biology, University of York, UK
[2] Department of Computer Science, Deramore Lane, University of York, UK
[3] Department of Electronics, Heslington, University of York, UK
[4] Immunology Unit, Institute of Molecular Medicine, University of Lisbon, Portugal

Peyer's Patches are secondary lymphoid organs which play a vital role in facilitating an immune response within the intestine. These form during embryonic development. Interestingly these patches develop randomly along the length of the mid-gut, varying in size, location and number. With traditional experimental techniques failing to explain this variability, we propose that by integrating available in vivo imaging data, gene expression analysis and gene knock-out data into an in silico model, we can accurately simulate lymphoid tissue formation in the mouse, permitting an exploration of the molecular, cellular and biophysical mechanisms involved.

Our poster describes how we have applied the CoSMoS framework in our approach, through the development of the domain and platform models in collaboration with experimental immunologists to implementation of an agent-based simulation and validation against available laboratory results. We show initial results revealing there is no statistical significant difference between the cell behaviour observed experimentally with that replicated by the model. With this assured, we then examine how the model captures the behaviour that emerges from cellular interactions, the formation of Peyer's Patches. We show that our model is more efficient at producing patches than has been seen in vivo. Although the model therefore does not correctly capture the observed emergent behaviour, we now have the opportunity to make use of the model to investigate the discrepancy. With the model created using such a principled framework, the abstraction can be examined in collaboration with domain experts in the hope of determining if this can be explained by the necessary inclusion of assumptions or by a gap in our biological understanding.

Harnessing emergent properties in artificial distributed networks: an experimental framework

George Eleftherakis[1], Ognen Paunovski[1], Konstantinos Rousis[1], and Anthony J. Cowling[2]

[1] South-East European Research Centre,
24 Proxenou Koromila, 54622 Thessaloniki, Greece
{geleftherakis,ogpaunovski,konrousis}@seerc.org
[2] Computer Science Department, University of Sheffield
Regent Court, 211 Portobello, Sheffield, UK
A.Cowling@dcs.shef.ac.uk

In many domains centralized architectures have been replaced by decentralized approaches which are able to offer significant advantages in utilization of network resources. Nevertheless, the increased demand and complexity of applications and services operating within distributed environments has demonstrated the need for more efficient, robust and adaptive solutions which will operate without manual configuration and management. Systems are getting increasingly complex and we will inevitably reach the point where humans will not be able to cope with it; both due to resource and capability limitations. It is for such situations that humans turn to nature for inspiration. A multitude of useful, typically self-*, properties can be observed to emerge by natural complex systems such as ant colonies and bird flocks. The ability to guide and use these emergent processes could prove very valuable since emergence is considered to be the basis for a variety of very useful phenomena including self-organization, self-optimization, adaptation as well as other beneficial properties encountered in complex systems. Incorporating such behaviours in artificial distributed networks (ADNs) could offer significant benefits to the development and performance of the system, making it highly available, scalable and robust. Having even partial control of this process, however, proves to be extremely difficult.

During the last few years there is an ongoing effort, from both Complex Systems and Multi-Agent Systems research communities, to tackle the problem of engineering emergence either in general or situated in a specific domain. Fromm [1] claims that the problem of engineering emergence equals to the problem of science in general. He proceeds with applying a variety of classical scientific methods on this problem and reaches the conclusion that an "intelligent design" based on the classic scientific

method is needed [1]. Stepney, Polack and Turner [2, 3] approach emergence engineering via rule migration, as a means of translating high level designs to low level implementations which exhibit emergent behaviour via upward causation. Welch *et al* [4] have presented promising results on emergence engineering by using basic engineering principles and an innovative computational architecture based on occam-π.

We believe that generic claims of engineering emergence are impossible to make and we doubt that a generic "one size fits all" solution can or will be provided soon, if ever. We propose the study of each specific domain separately, in an attempt to identify generic patterns and guidelines which could assist on introducing desired emergent properties. Previous work on our framework for studying emergence in a disciplined way, allowed us to gain insight of the causal relations between microscopic and macroscopic levels. In a second phase we aim to develop systems which exploit desired emergent properties, focusing on ADNs, aiming in the long term to provide designers and engineers with tools, models and patterns. Towards this direction and in an attempt to put forward several ideas on how to approach and revise engineering practices when dealing with systems which exhibit emergence, we devised an abstract process which we followed throughout our experimentations. Figure 1 depicts our experience on engineering ADNs which exploit emergent phenomena, as a successor to our framework for studying emergence. It is an abstract and laborious process of experimentation, attempting to provide a disciplined approach on harnessing emergent properties as a means of providing desired behaviours to complex systems.

We define as a starting point the analysis of a system to be built and the separation of the desired properties of that system in two categories: the first includes functionality we want to develop by following standard (software) engineering processes and methodologies and the other consists of desired properties which are considered to be related with emergent phenomena (e.g. self-adaptability). The next step consists of modelling the so called "normal" behaviour as a multi-agent system (MAS). The abstract model can be afterwards refined to a more formal and less ambiguous model such as an extended FSM. At this stage, the use of validation and verification techniques is strongly encouraged. Proper verification of the standard system model can ensure at later stages that the presence or absence of particular emergent properties is the effect of the "planted", emergence design, and not due to bugs and design flaws on the standard model.

The next step is to design, through a process of experimentation, the additions and modifications to the MAS model, which may lead to the appearance of the desired emergent phenomena. As already argued,

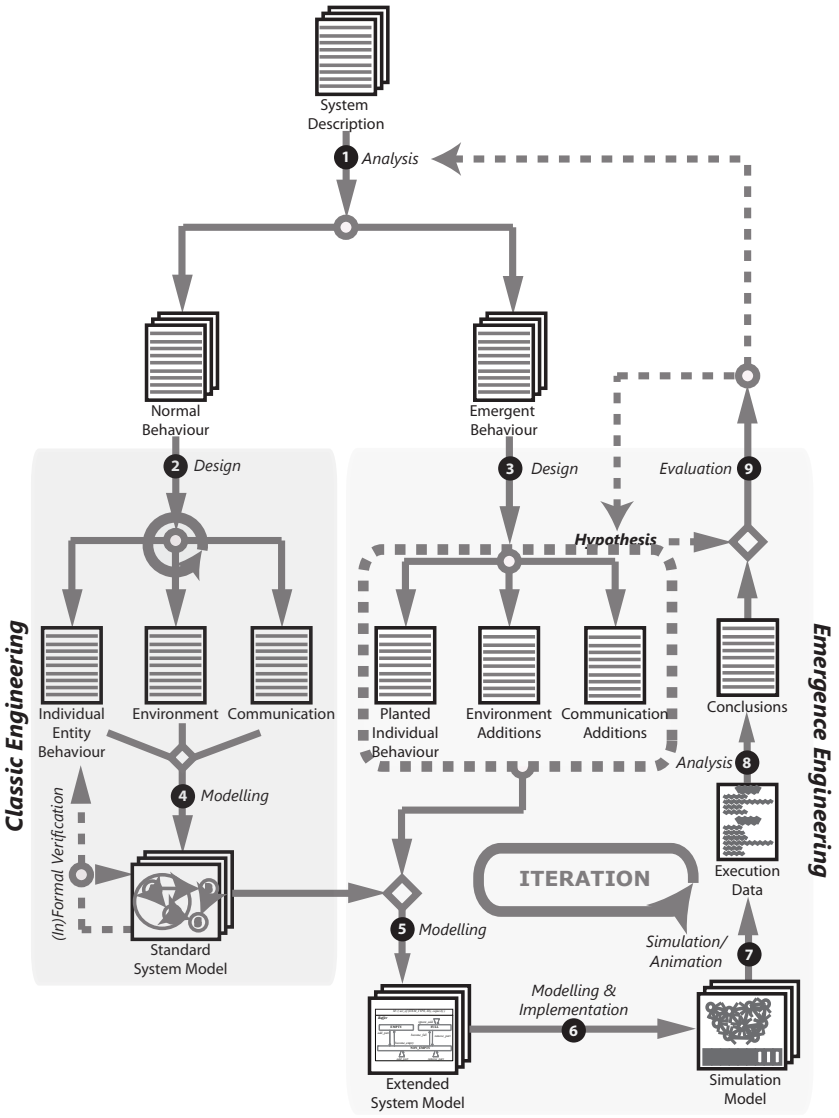


Fig. 1. An abstract process for engineering emergent properties in complex systems.

there cannot be strict or specific guidelines for this step; depending on the domain and the type of emergence expected, different patterns can be utilized to form an initial hypothesis. These additions, combined with

the standard model, form the *extended system model*. This model is then refined, translated or implemented in order to serve as a simulation model on which experiments can be run. After simulating and analyzing the results we have to reach some conclusions regarding our hypothesis. The last step consists of comparing and evaluating the conclusions against the initial hypothesis. At this stage, there is either a conjecture that the hypothesis is sound so the desired functionality can be implemented or the hypothesis is disproved in which case another iteration starts at step 5.

Currently, we are in an ongoing phase of experimentation, following the proposed framework, whereby we attempt to refine our hypotheses and conjecture them with supporting simulation results. The core of our hypothesis is that by introducing biologically inspired properties in an ADN we will be able to meet global level, non functional, requirements such as availability, adaptability, and robustness, among others.

References

[1] Jochen Fromm. On engineering and emergence. Technical Report nlin.AO/0601002, Distributed Systems Group, Kassel University, Jan 2006.

[2] Susan Stepney, Fiona Polack, and Heather Turner. Engineering emergence. In *ICECCS '06 Proceedings of the 11th IEEE International Conference on Engineering of Complex Computer Systems*, pages 89–97, 2006.

[3] Heather Turner, Susan Stepney, and Fiona Polack. Rule migration: Exploring a design framework for emergence. *International Journal of Unconventional Computing*, 3(1):49–66, 2007.

[4] Peter H. Welch, Kurt C. Wallnau, and Mark Klein. Engineering emergence: an occam-pi adventure. In *CPA 2009 Proceedings of the 32nd Communicating Process Architectures Conference*, 2009.

Simulating cellular automata using Go

Sarah Clayton, Neil Urquhart, and Jon Kerridge

School of Computing, Edinburgh Napier University,
Merchiston Campus, Edinburgh EH10 5DT.

Abstract. In this poster we discuss the implementation of Conway's Game of life, written in Google's Go programming language. Although this lacks many of the facilities of established concurrent CSP based languages, enough are available to create a simple cellular automata simulation running tens of thousands of concurrent processes correctly and robustly. The Game of Life is a well known cellular automaton. Its implementation provides a useful means of comparing Go to other languages.

1 Introduction

The Go programming language [6] was released by Google in 2009. To summarise the authors, its stated aims are: to provide simple concurrency using coroutines and channel communication; to maximise the potential of multicore processors and network distributed systems; to compile to machine code while providing garbage collection and reflection facilities [4].

Go's history is described in [9]. Its ancestors are Bell Lab's Newsqueak concurrent language, which inspired further concurrent languages such as Alef and the Limbo language running on the Inferno operating system.

Many of the constructs available in occam-pi [1] and JCSP [10] are missing in Go, however in this paper we will show that simple simulations such as Conway's Game of Life [3] can still be very simply implemented, and easily scaled upwards.

2 Language architecture

Like occam-pi, Go uses a lightweight thread system to run processes [7]. Go assigns a number of coroutines (termed goroutines) to a set of threads, and then automatically moves them to another, runnable,

thread should one of them block for any reason. Each goroutine is assigned a stack of a few kilobytes for memory, with more allocated from a segmented stack at runtime if needed. Stack segments are allocated and freed automatically. This allows for the creation of hundreds of thousands of goroutines, without the resource costs and programming complexity inherent with threads, using a memory model that also allows for automatic garbage collection [5].

There are no direct equivalents with CSP semantic constructs in Go. To an extent some Go language features can stand in for some of these constructs, but are not completely analoguous. A brief overview is given in Table 1:

occam-pi	**Go nearest equivalent**
Barriers	WaitGroup
Protocols	no
Channel ends	no
Channels	Channels are first class objects but implicitly shared
Parallel composition	no - each process is launched individually using the go keyword
Choice	select statement
Prioritised choice	no

Table 1. Comparison between occam-pi and Go language facilities

WaitGroups provide some synchronisation facilities, but the synchronisation operation is not atomic. This creates the risk of deadlock, requiring extra care in their use. Choice between ready guards is implemented using the select statement, that uses a pseudorandom choice mechanism. This is meant to ensure fairness on individual choices but makes it harder to guarantee fairness for the system as a whole.

3 Implementing phased synchronisation

Despite these disimilarities, it is possible to implement simulations similar to those created using occam-pi, using phased synchronisation [2] and the client-server architecture [12].

The client-server architecture is defined by the relationship between a client, in this example a cell in the Game of Life, and its server. This is implemented using channels that are built in at the heart of Go. All communications are initiated by the client, the server only responds to requests. This ensures the execution is deadlock free [11]. The phased

synchronisation where all agents discover their environment, blocking on a barrier until all have completed, before modifying their state, is implemented using the WaitGroup object.

Although with less sophisticated methods than those described in [8], the Game of Life was implemented with a fixed number of servers, acting independently of each other, and in charge of a set number of cells. Each cell has a channel connection to its local server. During the discovery phase, it requests information about the state of its neighbours from the server which replies with the sum of living cells adjacent to it orthogonally and diagonally. During the modify phase, the cell decides if it should be born if dead, survive if alive, or die, contingent on the number of living neighbours. After a cell updates its state, it communicates this to its server and waits on the modify barrier until all other processes have completed. This cycle is then repeated. It was possible to create a 100x100 grid, nearly 12,000 Goroutines including servers, that worked correctly (albeit slowly). This demonstrates that Go provides the same easy scalability as other CSP based languages. Visualisation was done using bindings for the GTK library, although OpenGL bindings are also available.

4 Conclusion

From this initial work we conclude that, as a process oriented programming language, spatial simulations can be robustly and scalably implemented with Google Go, despite the absence of many useful CSP constructs in the language. Future work will include experimentation with network distributed processes using the built in network channel communication facilities provided by the netchan package.

References

[1] F.R.M. Barnes and P.H. Welch. Kent retargettable occam compiler (kroc), 2009. http://www.cs.kent.ac.uk/projects/ofa/kroc/.

[2] F.R.M. Barnes, P.H. Welch, and A.T. Sampson. Barrier synchronisation for occam-pi. In H.R. Arabnia, editor, *Proceedings of the 2005 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'05)*, pages 173–179. CSREA Press, 2005.

[3] M. Gardner. Mathematical games: The fantastic combinations of john conway's new solitaire game life. *Scientific American*, 223(4):120–123, 1970.

[4] Go Authors. The go programming language. http://golang.org/.

[5] Go Authors. Why goroutines instead of threads? http://golang.org/doc/go_faq.html#goroutines.

[6] R. Pike. Go, 2010. http://www.oscon.com/oscon2010/public/schedule/detail/15464.

[7] C.G. Ritson, A.T. Sampson, and F.R.M. Barnes. Multicore scheduling for lightweight communicating processes. In J. Field and V.T. Vasconcelos, editors, *Coordination Models and Languages, 11th International Conference, COORDINATION 2009, Lisboa, Portugal, June 9-12, 2009*, pages 163–183. Springer, 2009.

[8] A. Sampson, P.H. Welch, and F. Barnes. Lazy cellular automata with communicating processes. In J.F. Broenink, H.W. Roebbers, J.P.E. Sunter, P.H. Welch, and D.C. Wood, editors, *Communicating Process Architectures 2005*, pages 165–175. IOS Press, 2005.

[9] A.T. Sampson. *Process-oriented patterns for concurrent software engineering*. PhD thesis, University of Kent, 2008.

[10] P.H. Welch, N.C. Brown, J. Moores, K. Chalmers, and B. Sputh. Integrating and extending jcsp. In A.A. McEwan, S. Schneider, W. Ifill, and P. Welch, editors, *Communicating Process Architectures 2007*. IOS Press, 2007.

[11] P.H. Welch, G.R.R. Justo, and C.J. Willcock. Higher-level paradigms for deadlock-free high-performance systems. In R. Grebe, J. Hektor, S.C. Hilton, M.R. Jane, and P.H. Welch, editors, *Transputer Applications and Systems '93, Proceedings of the 1993 World Transputer Congress*, pages 981–1004, 1993.

[12] P.H. Welch, B. Vinter, and F. Barnes. Initial experiences with occam-pi simulations of blood clotting on the minimum intrusion grid. In H.R. Arabnia, editor, *Proceedings of the 2005 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'05)*, pages 201–207. CSREA Press, 2005.