The University of York Department of Computer Science

Submitted in part fulfilment for the degree of MEng.

Parallel Programming Tools for Exploring Immune System Development

Oliver Binns

23rd April 2018

Supervisor: Dr. Fiona Polack, Dr. Kieran Alden

Number of words = 6825, as counted by `wc -w`.
This includes the body of the report only.

Abstract

More powerful computers are paving the way for sophisticated simulations of previously underexplored complex biological systems. Agent-Based Modelling has been previously used to implement this type of simulation, showing how the behaviour of a number of individual agents can contribute to the system as a whole.

As advancements in computing tend towards parallelism and distributed systems, these increasingly elaborate simulations must take full advantage of this change in order to be computed in a reasonable time period. Creating programs and simulations that make efficient use of this parallelism can be technically challenging. In particular it requires implementing an optimal distribution of tasks over the system and ensuring the integrity of any memory that is shared across multiple threads. Some significant problems, including a substantial computer science skills shortage, are slowing down the rate of progress in this area.

In this project the FLAME GPU framework is used to help mitigate some of the difficulties with implementing parallel programs, in order to build on an existing sequential MASON simulation. During the building of the simulation, I have discovered that it is not trivial to compare between the old and new implementations, due to differences in the frameworks used. For this reason, the final comparison will be done directly to the domain model itself.

Finally, I also propose Model-Driven Engineering (MDE) as a potential short-term solution to the computer science skills shortage, exploring some of the current MDE technologies for creating simulations and proposing further advancements to these tools for the future. These engineering techniques help to clearly show how the simulation is derived from the initial biological domain model.

Acknowledgements

I would like to my supervisors, Fiona Polack and Kieran Alden, for their support and guidance throughout this project. I would also like to thank the FLAME GPU development team at the University of Sheffield for their help in utilising the latest and pre-released features of this framework.

Contents

List of Figures

List of Tables

Acronyms

ABM Agent-Based Modelling. 3, 13, 18, 19

CoSMoS Complex Systems Modelling and Simulation Infrastructure. 21

CPU Central Processsing Unit. 18

CUDA Compute Unified Device Architecture. 18

EVL Epsilon Validation Language. 23, 31

FLAME Flexible Large Scale Agent Modelling Environment. 19

FLAME GPU FLAME for the GPU. 3, 5, 19–24, 30, 31

GPGPU General Purpose GPU programming. 11

GPU Graphics Processing Unit. 11, 18, 19, 21

LTi Lymphoid Tissue inducer cell. 22

LTin Lymphoid Tissue initiator cell. 22

LTo Lymphoid Tissue organiser cell. 22

MASON Multi-Agent Simulator Of Neighborhoods. 3, 19, 22, 24

MDE Model-Driven Engineering. 3, 17, 19

Glossary

FLAME GPU An extension of the FLAME framework which uses GPUs for parallelism. 3, 5, 19–24, 30, 31

LTin A moving cell that can bind to any static cell. Reponds to chemokine expression. 22

LTi A moving cell that can bind to any static cell. Believed to be responsible for initiating Peyer's Patch development. 22

LTo A static cell. Differentiates (divides) on stable bind with an LTin and LTi cell. 22

MASON A Java framework for producting simulations using ABM. 3, 19, 22, 24

Host In GPGPU programming, the computer containing the GPU is referred to as the host. 22

in-silico Scientific tests conducted using computer simulation. 10

in-vitro Scientific tests conducted using real biological systems under laboratory conditions. 10, 11

Peyer's Patch Clusters of cells that form in the small intestine. 11, 21

1 Introduction

Agent-Based Modelling (ABM) has been successfully used to simulate biological systems and develop new scientific knowledge[1], [2]. While traditional mathematical-based models are generally much more efficient to compute and produce good results when exploring how biological factors effect populations as a whole, ABM is particularly useful when considering how individuals and their interactions produce emergent system behaviour. Mathematical-based models are limited in this aspect as they assume that each individual within the population is identical. ABM explicitly represents each individual, meaning these can be independent, with state and attributes, and fully autonomous. Unfortunately, these agent-based simulations require significantly more compute resources, thus taking a long time to run, particularly as simulations become more and more complex.

Here, and throughout this report, the term complexity does not necessarily refer to simulations that contains several components that are difficult to engineer. Rather, we discuss simulations that are made up of lots of simple components that interact with each other and the environment in such a way that the system behaviour is so elaborate that it can only be predicted by running the simulation.

As these simulations often contain large quantities of non-deterministic behaviour, numerous runs are required to ensure that results are statistically significant. Consequently, it is important to ensure that the simulations make efficient use of the available of the available compute resources in order to gather these results in a reasonable amount of time. This is one of the most significant constraints upon the current use of agent-based simulations within biological research.

Another constraint affecting the adoption of ABM simulations is the ease with which they can be created. At present, domain experts with knowledge of a particular biological system produce domain models of the system. These domain models are then converted into platform models, with the aid of software developers. The domain experts verify that the platform models are reasonable abstraction of the system, and they are implemented in code by the software developers. This is the

CoSMoS process (Fig. 1.1).

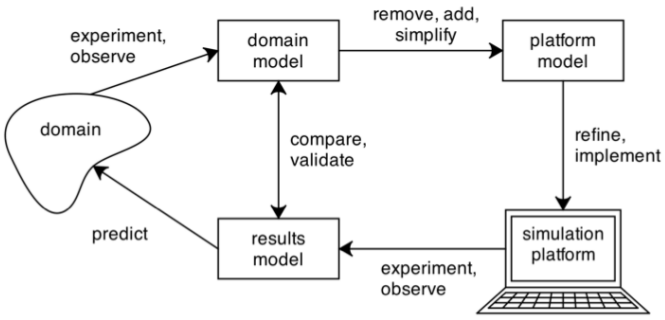


Figure 1.1: CoSMoS Process (Taken from [3])

The skills shortage is the current bottleneck in the adoption of the CoSMoS process. If the development process can be automated by creating general solutions for common biological processes, such as cell division, it will allow simulations to be created more easily.

1.1 Motivation

The motivation behind this project is the new knowledge that could be gained from the ability to create efficient parallel biological simulations more easily. While many biological systems can not be fully studied in-vitro, simulations can provide a valid alternative. Indeed previous work has been used to produce novel biological hypotheses which have been shown to be statistically similar to those observed in the laboratory[1, p.174].

However, these previous simulations of biological systems have had to make significant trade-offs in their implementation. For example, 3D environments have had to be mapped into 2-dimensions, while these additional abstractions may not invalidate the model..?

Work in this area is needed to help reduce the number of trade-offs needed when designing platform models for simulations.

It is hoped that, in the future, this type of in-silico experimentation could significantly improve the process by which drugs are developed. Extensive in-silico testing can be performed much more quickly than the current in-vitro testing, reducing the time it takes to research new drugs and cures. Drugs and cures would be available for patients earlier, saving many lives. Additionally, the significant financial burden of

drug development, which is estimated to be around $2.9bn[4], could be reduced, freeing up heavily contested funds for additional research. Finally, the use of animal testing could be reduced and, in the long term, eliminated completely.

1.2 Project Aims

In summary, the aims of this project are to

1. Establish a firm grounding for the future development of new tools to allow fast, parallel simulations of biological systems to be easily created by non-technical users.
    a) Explore the findings of the new implementation and discuss how these can be generalised to new simulations.
    b) Discuss techniques for allowing non-technical users to easily create formal models that can be transformed into new simulation implementations.
2. Develop a **parallel** implementation of an existing **sequential** simulation of Peyer's Patch development and explore any speed increases that can be produced using General Purpose GPU programming (GPGPU) programming.
3. Review of the use of simulations, with a particular focus on computational biology. This review should explore the advantages that in-silico testing provides over in-vitro experimentation and the problems that must be overcome for the mass adoption of biological simulations.

1.3 Statement of Ethics

This project was conducted in accordance with the University of York's code of practice on ethics. This project does not involve human participants, so guidelines on informed consent and confidentiality will be met. No confidential medical data or personal information has been used during the course of the project development. This project has involved no direct animal participation.

The simulation of the biological model is for the purpose of developing understanding of applying GPGPU methods to an agent based model of

a biological system. It will not be used its current form to publish novel biological findings and does not fully simulate a biological process.

1.4 Report Structure

This report details the work done throughout the project and

1. Chapter 2 gives an overview of simulations and the benefits and limitations of their use particularly with regard to computational biology.
2. Chapter 3 details the development of an improved, inherently parallel, implementation of PPSim (PPSim v2) and the MDE tools that were developed to aid this work.
3. Chapter 4 evaluates the progress that this project has made towards making **fast**, parallel simulations more **accessible**.

2 Literature Review

2.1 Simulation

Simulations are model-based imitations of a system which feature its key characteristics and behaviours. Computer simulations are used in a wide range of disciplines on applications such as video games, medicine, product development and even nuclear weapons.

The models of systems used in simulations may have a varying amounts of abstraction. Simulations used for teaching will likely have models which remove significant amounts of complexity from the system. Removing complexity from the original system can also help to reduce the time it takes to run the simulation, a key focus of this project. Simulations used for video games tend to be as realistic as possible as realism has been shown to produce a higher level of immersion[5], a highly desirable attribute of games.

In particular this project will focus on simulations created using ABM. ABM is a technique that explores the autonomous behaviour and interaction between a number of individuals, known as agents. The aim of this is to show how the individuals' behaviours interact to produce the system as a whole. This is in contrast to more traditional mathematical top-down models which consist of a set of equations which establish relationships between a set of variables.

This chapter begins by outlining the benefits and current limitations of using simulations generally. It then discusses some potential solutions to some of these limitations before applying this to some of the existing simulations that have been developed.

2.1.1 Benefits

Feasibility

Exploring computer simulations is can be far more feasible than exploring a real world environment. Video games are simulations which may allow players to experience scenarios that they may not otherwise get the

opportunity to encounter. For example, car racing games are significantly cheaper and safer than real life racing.

Real-world scientific testing may not be feasible for a number of reasons. Simulating the aerodynamics of new car designs virtually is far quicker and cheaper than creating multiple different prototypes for physical tests. Morality may be a factor, animal testing for cosmetic products or medicine is a good example of this. With nuclear weapons, legality is a key issue, as some weapon testing is banned under a number of global treaties[6], [7]. Finally, some real-world tests may be too dangerous to perform, such as in the case of invasive medical examinations. In all of these examples, computer simulation is routinely used to reduce or replace real-world testing.

Reducing Complexity

Reducing complexity through abstraction allows better understanding of the system to be gained as the complexity may initially be overwhelming.

Environmental Control

Computer simulations allow for the environment to be more easily controlled. The ability to adjust external factors and independent variables that may affect the system on demand can be particularly useful. This ability can be used for illustrating why different variables in system processes are important.

Time can be manipulated to visualise system processes at more reasonable time scales. A chemical process that takes a fraction of a second can be slowed down to ensure that it can be seen.

Simulations allow for additional tests to be easily added at a later date. If the researcher wants to discover how an additional variable is related to system behaviour, it can be added and the simulation can be easily re-run.

Visualisation

One of the biggest benefits of simulation is the ability to graphically visualise a system or its constituent parts. Using simulation for visualisation has number of benefits over attempting to demonstrate real world systems. Several of these benefits translate from the previous two

sections- they can be more feasible to explore than a real world environment and featuring a reduced complexity can allow the key concepts to be understood without overwhelming the user.

Simulation can be particularly useful for visualising concepts for education.

Particularly immersive visualisations can also be produced using virtual and augmented reality. The Virtuali-Tee is an educational tool that uses simulation and augmented reality to provide a view at the body's internal organs[8]. Little Journey aims to reduce kids' anxiety about surgery by providing a realistic tour of their hospital ward given by animated cartoon characters[9].

2.1.2 Limitations and Constraints

While simulations seem very useful across a wide number of fields, there are some significant limitations as to where and how they can be used.

Insufficient Domain Knowledge

A simulation is based on a model of a system. A model is an abstraction from reality representing only the necessary key characteristics and behaviours of the system. In order for the model to the be faithful to its original domain it must be able to produce demonstrably similar behaviour from similar system components.

A lack of knowledge regarding the domain of the simulation is one of the most significant constraints regarding its implementation. If this is the case, the model produced may be incorrect or abstractions may remove necessary detail required for the system to function as expected. Unfaithful models can also be a result of *incorrect* knowledge of the domain. For complex systems, having too many abstractions from the original domain may also invalidate the model and produce incorect results[10, p.8].

While knowledge of a system needs to be sufficient to produce reasonable assumptions, it need not be perfect. This is particularly the case with the type of simulation discussed later in this report. In this case, assumptions about the domain are made

Compute Power

Complex models with too few abstractions from their domain may require significant computing power to simulate. Additional abstractions may not be possible as they may invalidate the model. In these cases, cutting edge hardware may be needed for the simulation to be run in an acceptable time.

Powerful hardware is expensive to access, so this may be a significant constraint on the ability to simulate.

Skills Shortage

The previous section discussed reliance that complex simulations have on advanced hardware. However, advanced hardware alone will not necessarily allow a simulation to compute in a reasonable amount of time. With modern architectures which are becoming increasingly parallel, the simulation code must be tailored to take advantage of the computing power available. Efficient parallel programs that do this rely on the availability of experienced programmers. May recent studies have highlighted the existance of significant computer science skills shortages, across the world[11], [12]. These skills shortages may be significantly limiting the possibility for cross-disciplinary work to utilise fast, advanced simulations.

Bugs

As with any form of computer program, mistakes can be made causing bugs to be present in the simulation code. Bugs may cause the simulation to be incorrect meaning any hypotheses and results are based on incorrect data.

This is linked to, but not the same as, having insufficient domain knowledge. Both of these limitations will cause the simulations fail silently, produce incorrect results with no immediately obvious failure[CITE]. However, while these problems are specific to simulation, they are not dissimilar from the issues that can occur from poorly designed real-world testing.

If the simulation needs to be safety-critical, developing it using formal methods and refinement may be a good way to ensure that no bugs are introduced in the code.

2.2 Improving Simulations

Significant constraints on when simulations can be used were discussed in the previous chapter. In order for simulations to become more widely adopted, some of these constraints must be overcome. In particular, this project focuses on two particular sections of which I believe will have the biggest immediate impact.

Firstly, attempting to overcome the problems provided by the current computing skills shortage. Initial work to help overcome the skills shortage will likely come by providing tools that can automate work done by technical users in order to increase their productivity. Eventually, if this technical work can be fully automated, non-technical users will be able to easily create advanced simulations.

Secondly, simulations need to be able to generate results in a realistic amount of time. As many biological simulations include significant elements of randomness, they must be run a large number of times in order to produce reliable results.

This chapter discusses the potential solutions to each of these problems.

2.2.1 Ease of Creation

Ease of creation (and maintenance) is an important feature for future simulation particularly due to the aforementioned computer science skills shortage (Section 2.1.2).

Many initiatives are being rolled out to address this issue. Computing was added to the National Curriculum in 2013[13], [14] and numerous technology companies are also creating their own campaigns[15], [16]. It is likely that these will take a number of years to yield results, but should eventually mean that even domain-experts have a basic grasp of code.

Proprietary Solutions

Some proprietary solutions for simulation creation are available.

One example of this, with a resonable level of industry support, is Biocellion. While it is freely-available for non-profit use, it is still not fit our requirements of being simple-to-use. It requires a level of proficiency with the Linux OS, C++ programming and familiarity with mathematical modelling concepts. [17]

Model-Driven Engineering

This project has already discussed abstractions with regards to modelling for simulations. Abstractions are also particularly useful in software engineering. As software gets more complex, addition abstraction is generally needed in order to extract the important details of the implementation[18, p.24].

MDE is a software abstraction for creating and exploiting domain models, such as those that have been previously discussed with regards to simulations. MDE has shown a promising increase in understanding between stakeholders and can produce productivity gains when models are reused across projects[19].

Flexible Modelling

Flexible Modelling tools could be a good method for allowing new simulations to be created more easily. Using flexible modelling, non-technical users would be able to create sketches which can be automatically processed by tools into formal models and prototype metamodels[20]. FlexiSketch is a good example of this and provides a good tool for creating models and metamodels for software development[21].

2.2.2 Speed Up

Machine Learning

A solution to the speed problem that has been proposed recently is to use machine learning on a small set of results to produce[22] This has problems in that... likely affected by Standard Machine Learning issues? Bias? Overfitting?

Already an ongoing area of research..

Parallelism

Parallelism fundamentally changes the game and allows computers to keep following Moore's law even has engineers are struggling to make transisters ever smaller and smaller[23]. As modern computers tend further towards parallelism to keep providing the speed-ups that have been inherent in the industry over recent years, new parallel algorithms need developing in order to take full advantage of the computing power available.

Distributed Systems MapReduce is a programming model that is generally used for processing big data in parallel using distributed systems.

CPU vs GPU Modern computers provide two main methods for parallelising code. We are building on a previous project[24] which layed the groudwork for this. This previous project outlined the choice between CPU and GPU parallelism and makes the case for exploring GPUs- simplisitically put, this is due to the significantly greater speed ups that can be achieved. Indeed it has been shown that GPU simulation on a standard desktop computer can easily produce performance rates that are better than those of a high-performance computing cluster[25]. One case study even reduced the time taken for a simulation to be computed from hours on a CPU to just seconds on a GPU[2].

Multiple CPU cores are now commonplace in modern computers, including smartphones. GPU normally only found in gaming PCs and consoles This is changing, Apple now includes a custom GPU in many of its mobile devices- metal

OpenGL vs CUDA A previous project compared OpenGL and CUDA in detail.

2.3 Existing Simulations

2.3.1 Agent-Based Modelling

As previously mentioned, ABM models the autonomous behaviour of individual agents and the interaction between them. Agents are often confused with objects, a commonly-used programming concept. However, it's important to emphasize the differences between them. In particular, while both agents and objects recognise the importance of interactions, objects are totally obediant to their method calls whereas agents have autonomy. This means that agents can interact and react to communications according to their own agenda.

Custom Code?[24] not well tested less easy to update to support new tools and hardware, new CUDA GPUs Each custom code simulation must be updated separately

While, Agent-Based Modelling is more computationally expensive than top-down mathematical modelling, it is also more natural to model and intuitive to parallelise[25]. Communication between individual agents

can be difficult to implement in parallel but parallel communication is by no means limited to ABM.

Mention FLAME (traditional) is an attempt to make simulations more accessible via ABM.

MASON

MASON is a multi-agent simulation library for Java. MASON has been previously evaluated as the one of the fastest sequential ABM libraries available[26], however this evaluation took place over a decade ago and does not factor in any of the recent work into GPU parallelism.

FLAME GPU

FLAME for the GPU (FLAME GPU) is an extension of the original version of FLAME, where the simulations that are created are compiled down to parallelised CUDA code. This means the simulation can take full advantage of the significant power of modern NVIDIA GPUs. As its name suggests, FLAME GPU uses Agent-Based Modelling to implement simulations.

While relying on a framework such as FLAME GPU can cause problems if support is stopped, FLAME GPU is an open source project, so this should slightly be less of a concern.

model based -> FlameGPU[2] restricted to the framework limitations –step function host-based agent creation

The FLAME GPU framework brings its own advantages. As it maps a model to simulation code, it brings with it many of the advantages of Model-Driven Engineering. In particular, it gives domain experts with limited programming knowledge the ability to understand and help design models. Additionally, the auto-generation of artifacts, such as simulation code templates, significantly lowers the boundary to entry to less advanced programmers. Finally, As FLAME is continuously updated to support new hardware advances, such as multicore GPU architectures[25], the implementations can always be certain of taking advantage of the latest hardware optimisations and portability features that FLAME provides. Existing simulation models that are implemented on top of FLAME GPU may receive these performance improvements without major code changes.

FLAME GPU takes input in the form of an XMML model which defines the types of agents that feature in the simulation and their interactions

FLAME GPU also provides a message passing interface which abstracts away from the underlying agent communication. This means that programmers need not be concerned with communication between threads which is one of the most difficult tasks in parallel programming.

2.3.2 Biological Simulations

Computational Biology is a relatively new field of study that has been growing signficantly over the last decade[CITATION NEEDED].

Within Biology, simulation is often required as an alternative to invasive medical testing/animal testing

Simulations have even been proposed as a method for exploring a potential set of first principles and mathematics that are specific to biology which could even constitute a new subject- theoretical biology[27].

Genome

Cell Dynamics

2.3.3 PPSim

This project focuses on simulation as a tool for exploring biological systems at cell level. It uses the existing simulation of Peyer's Patch[1] and attempts to use parallel computer architectures in order to speed this simulation up.

Finally, I propose a new tool, which builds on existing work in order to make this power available to non-technical users.

3 Methods

Having established the current difficulties with biological simulation, this chapter details the development of tools and simulation experiment performed as part of this project and how these difficulties are mitigated. It describes the implementation of a new GPU-based PPSim platform implementation and presents a new tool for mapping and transforming biological models into FLAME GPU simulations.

3.1 Tools

This project was developed on a machine running Ubuntu 16.04 with Intel i5 (650 @ 3.20 GHz) and 8GB of RAM. The GPU utilised was an NVIDIA GeForce GTX 1050. GPU programming utilised the latest version of CUDA (9.1).

In order to produce a working simulation of Peyer's Patch formation, I have used an unreleased version (1.5) of FLAME GPU. As FLAME GPU is developed openly, this version is freely available online[28]. FLAME GPU has been added to the project repository as a submodule, to ensure version compatibility between FLAME GPU and PPSim v2.

Eclipse Neon v4.6 and Epsilon v1.4 have been used with Java Runtime Environment 1.8.0 for developing the input XML model for FLAME GPU.

Version-control for both the simulation code and this report has been managed through Git and is available on GitHub. As this is an individual project, Git has been an acceptable solution. If the scope of the project is to expand, it may become necessary to move the modelling section of the development into a more suitable version-control system, such as EMFStore[29] which is specifically designed for models.

3.2 The Domain Model

The domain model is taken from the existing sequential simulation of Peyer's patch development described in Section 2.3.3. This model has been developed according to the CoSMoS process[10].

Peyer's patches are clusters of cells that form on the lining of the small intestine.

In this simulation there are three types of agent which are all types of cells; LTo, LTin and LTi. Stationary LTo cells

3.3 The Platform Model

In the platform model there is a single *active* LTo cell in the centre of the environment. There are no LTi and LTin cells in the environment when the simulation is initialised. A Host function manages cell migration into the environment by creating these cells in random positions at a constant rate.

A previous project on this topic stated that LTo cell division must be performed sequentially on a host function in order to prevent new cells occupying the same space[24]. However, by adding a force resolution step, as used in previous FLAME GPU simulations[2], this has been implemented in parallel, on the device.

3.4 FLAME GPU

Building FLAME GPU simulations requires the creation of two artifacts (Fig. 3.1), a structural 'XML Model File' which defines the agents in the system and their allowed channels for communication and a C source code file of 'Scripted Behaviour' which defines the actual behaviour of agents.

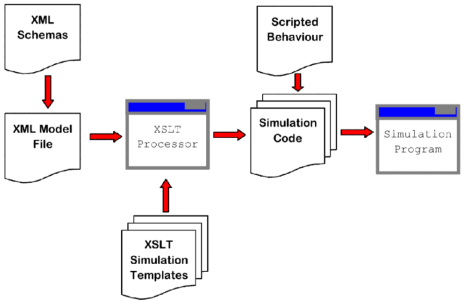


Figure 3.1: FLAME GPU Development Process (Taken from [25])

A state-based platform model design is already available- the domain model for the original MASON implementation of this simulation. As FLAME GPU uses state machines for each agent type in its implementation, these have been directly implemented into it.

Host cell creation into the system is a work in progress for next FLAME GPU release?

Due to time constraints, I have not been able to implement environmental growth.

3.4.1 Epsilon

In the creation of the FLAME GPU XML model, I decided to implement a new tool, built using the Epsilon Modelling Framework. This new tool contains a graphical editor which allows the FLAME GPU 'XML Model File' to be created diagramatically. If implemented correctly, the tool should *directly map* the platform model structures into the FLAME GPU domain model. This should significantly improve the model design experience and allow for domain experts to be kept more involved with the simulation creation. Figure 5.1a shows the Peyer's Patch simulation created within this graphical editor. The XML model used for the final simulation was automatically generated from this diagram.

The Epsilon Validation Language (EVL) has allowed well-formedness constraints to be implemented for the FLAME GPU input model. This should provide intuitive feedback (Fig. 5.1b) in cases where the model may be incorrect, in order to prevent an invalid XML model from being produced. Quick fixes (Fig. 5.1c) have also been implemented to guide the user through easily correcting their invalid model.

One final benefit of this new approach to model generation, is that it makes it far simpler to .

3.4.2 Testing

As the new simulation is a re-engineering of the existing PPSim, it can inherit the validation of the existing simulation. This requires the new simulation to be demonstrably similar with similar components, behaviours and results.

Talk about how the model was tested to ensure correctness
Mention missing link in (incorrect) model from Kieran's paper

4 Results and Evaluation

4.1 Findings

4.1.1 Ease of Creation

This proof of concept has shown that direct mappings from a biological domain model to a platform implementation are possible. Based on this, further work is now being developed with Paul Richmond's FLAME GPU group, to enhance the accessibility of FLAME GPU for simulation developers. This should allow the simulation developer to focus on getting their domain details correct, rather than battling the agent platform. An improved mapping, which abstracts away all implementation details, could bring additional benefits in this area, including ensuring that the simulation code always supports the latest hardware features. This work is ongoing.

4.1.2 Speed Up

The differences between the MASON and FLAME GPU platforms have meant that it is not trivial to compare them.

The initial hypothesis of the platform models being comparable is untenable.

Instead, a new hypothesis is that the platform model for the FLAME GPU implementation is comparable to the original biological model.

[30] could be important for evaluating the performance of FLAME GPU against original PPSim

4.2 Conclusion

4.3 Further Work

4.3.1 Peyer's Patch Case Study

While the two Peyer's Patch simulations are both derived from the same domain model, there are clearly significant implementation differences between these. Future work will need to explore the meaning of these simulations and whether that have any effect on the validity of the model.

4.3.2 Software Generalisibility

While the use of Epsilon has allowed for domain experts to be kept involved with the model implementation, there is still a final, most technically challenging, stage of the simulation creation where the agent behaviour is programmed. Further work will need to study agent behaviour in these forms of simulation generally. The reuse of a previous simulation's implementation of cell division behaviour, has shown that at least some behaviour is common across different simulations. The power of MDE is such that this repeated behaviour should be generalised to reduce the time needed and prevent the mistakes that occur during reimplementation of the similar software features.

On top of this, with the current implementation, technical implementation details (layer functions and agent partitioning) have become part of the model. Ideally these should be extracted from the model implementation and automated.

4.3.3 Hardware Availability

One of the greatest challenges of this project has been gaining access to GPU hardware. While these are available off the shelf in most high-street computer retailers, they are not commonly found as part of standard desktop PCs, which generally containing integrated graphics hardware. Indeed, none of the software lab PCs at the University of York contain the dedicated graphics chips required to run this software.

Bibliography

[1] K. J. Alden, '"Simulation and Statistical Techniques to Explore Lymphoid Tissue Organogenesis"', PhD thesis, University of York, Heslington, York, 2012.

[2] P. Richmond, D. Walker, S. Coakley and D. Romano, '"High performance cellular level agent-based simulation with FLAME for the GPU"', *Briefings in Bioinformatics*, vol. 11, no. 3, pp. 334–347, 2010. DOI: 10.1093/bib/bbp073.

[3] M. N. Read, '"Statistical and Modelling Techniques to Build Confidence in the Investigation of Immunology through Agent-Based Simulation"', PhD thesis, University of York, Heslington, York, 2011.

[4] J. A. DiMasi, H. G. Grabowski and R. W. Hansen, '"Innovation in the pharmaceutical industry: New estimates of R&D costs"', 2016, pp. 20–33. DOI: 10.1016/j.jhealeco.2016.01.012.

[5] K. Cheng and P. A. Cairns, '"Behaviour, Realism and Immersion in Games"', in *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '05, Portland, OR, USA: ACM, 2005, pp. 1272–1275, ISBN: 1-59593-002-7. DOI: 10.1145/1056808.1056894. [Online]. Available: http://doi.acm.org/10.1145/1056808.1056894.

[6] United Nations Office for Disarmament Affairs. (Oct. 1963). Partial Nuclear Test Ban Treaty, [Online]. Available: http://disarmament.un.org/treaties/t/test_ban (visited on 08/02/2018).

[7] U.S. Department of State. (Dec. 1990). Threshold Test Ban Treaty, [Online]. Available: https://www.state.gov/t/isn/5204.htm (visited on 08/02/2018).

[8] Curiscope Ltd. (). Curiscope, [Online]. Available: https://www.curiscope.co.uk (visited on 05/02/2018).

[9] Little Sparks Hospital Ltd. (). Little Sparks Hospital, [Online]. Available: https://littlesparkshospital.com (visited on 05/02/2018).

[10] P. S. Andrews, F. A. C. Polack *et al.*, '"The CoSMoS Process, Version 0.1: A Process for the Modelling and Simulation of Complex Systems"', 2010. [Online]. Available: http://www.cs.york.ac.uk/ftpdir/reports/2010/YCS/453/YCS-2010-453.pdf (visited on 19/04/2018).

[11] Ecorys UK, '"DIGITAL SKILLS for the UK ECONOMY"', Jan. 2016. [Online]. Available: https://www.gov.uk/government/publications/digital-skills-for-the-uk-economy (visited on 15/01/2016).

[12] Microsoft. (Dec. 2012). Investing in American Innovation and the Next Generation, [Online]. Available: https://blogs.microsoft.com/on-the-issues/2012/12/12/investing-in-american-innovation-and-the-next-generation/ (visited on 16/01/2018).

[13] Department for Education, UK. (11th Sep. 2013). National curriculum in England: computing programmes of study, [Online]. Available: https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study (visited on 19/04/2018).

[14] ——, (3rd Jun. 2014). Microsoft, Google and IBM to help train computing teachers, [Online]. Available: https://www.gov.uk/government/news/microsoft-google-and-ibm-to-help-train-computing-teachers (visited on 23/04/2018).

[15] Apple. (24th May 2017). Apple launches app development curriculum for high school and community college students, [Online]. Available: https://www.apple.com/uk/education/teaching-code/ (visited on 19/04/2018).

[16] Microsoft News Center. (16th Sep. 2015). Microsoft expands global YouthSpark initiative to focus on computer science, [Online]. Available: https://news.microsoft.com/2015/09/16/microsoft-expands-global-youthspark-initiative-to-focus-on-computer-science/ (visited on 23/04/2018).

[17] ().

[18] R. E. Bryant and D. R. O'Hallaron, *Computer Systems: A Programmer's Perspective*, 2nd ed. Prentice Hall, 2011.

[19] J. Hutchinson, J. Whittle and M. Rouncefield, '"Model-driven engineering practices in industry: Social, organizational and managerial factors that lead to success or failure"', *Science of Computer Programming*, vol. 89, no. Part B, pp. 144–161, 2014, Special issue on Success Stories in Model Driven Engineering, ISSN: 0167-6423. DOI:

https://doi.org/10.1016/j.scico.2013.03.017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167642313000786.

[20] R. F. Paige, A. Zolotas and D. Kolovos, '"The Changing Face of Model-Driven Engineering"', in *Present and Ulterior Software Engineering*, M. Mazzara and B. Meyer, Eds. Cham: Springer International Publishing, 2017, pp. 103–118. DOI: 10.1007/978-3-319-67425-4_7. [Online]. Available: https://doi.org/10.1007/978-3-319-67425-4_7.

[21] D. Wüest, N. Seyff and M. Glinz, '"FlexiSketch: a lightweight sketching and metamodeling approach for end-users"', *Software & Systems Modeling*, Sep. 2017, ISSN: 1619-1374. DOI: 10.1007/s10270-017-0623-8. [Online]. Available: https://doi.org/10.1007/s10270-017-0623-8.

[22] K. Alden, J. Cosgrove, M. Coles and J. Timmis, '"Using Emulation to Engineer and Understand Simulations of Biological Systems"', unpublished.

[23] H. Sutter. (Dec. 2004). The Free Lunch is Over: A Fundamental Turn Toward Concurrency in Software, [Online]. Available: http://www.gotw.ca/publications/concurrency-ddj.htm (visited on 08/02/2018).

[24] P. Drew, '"Parallel Programming Tools for Exploring Immune System Development"', Master's thesis, University of York, Heslington, York, 2017.

[25] P. Richmond and D. Romano, '"Template-Driven Agent-Based Modeling and Simulation with {CUDA}"', in *{GPU} Computing Gems Emerald Edition*, ser. Applications of GPU Computing Series, W.-m. W. Hwu, Ed., Emerald Edition, Boston: Morgan Kaufmann, 2011, pp. 313–324, ISBN: 978-0-12-384988-5. DOI: https://doi.org/10.1016/B978-0-12-384988-5.00021-8.

[26] S. F. Railsback, S. L. Lytinen and S. K. Jackson, '"Agent-based Simulation Platforms: Review and Development Recommendations"', *SIMULATION*, vol. 82, no. 9, pp. 609–623, 2006. DOI: 10.1177/0037549706073695.

[27] D. Noble, '"The rise of computational biology"', *Nature Reviews Molecular Cell Biology*, vol. 3, pp. 459–463, 2002. DOI: 10.1038/nrm810.

[28] FLAMEGPU. (). FLAMEGPU, [Online]. Available: https://github.com/FLAMEGPU/FLAMEGPU/branches (visited on 18/03/2018).

[29] (). EMFStore, [Online]. Available: https://www.eclipse.org/emfstore/ (visited on 17/11/2017).

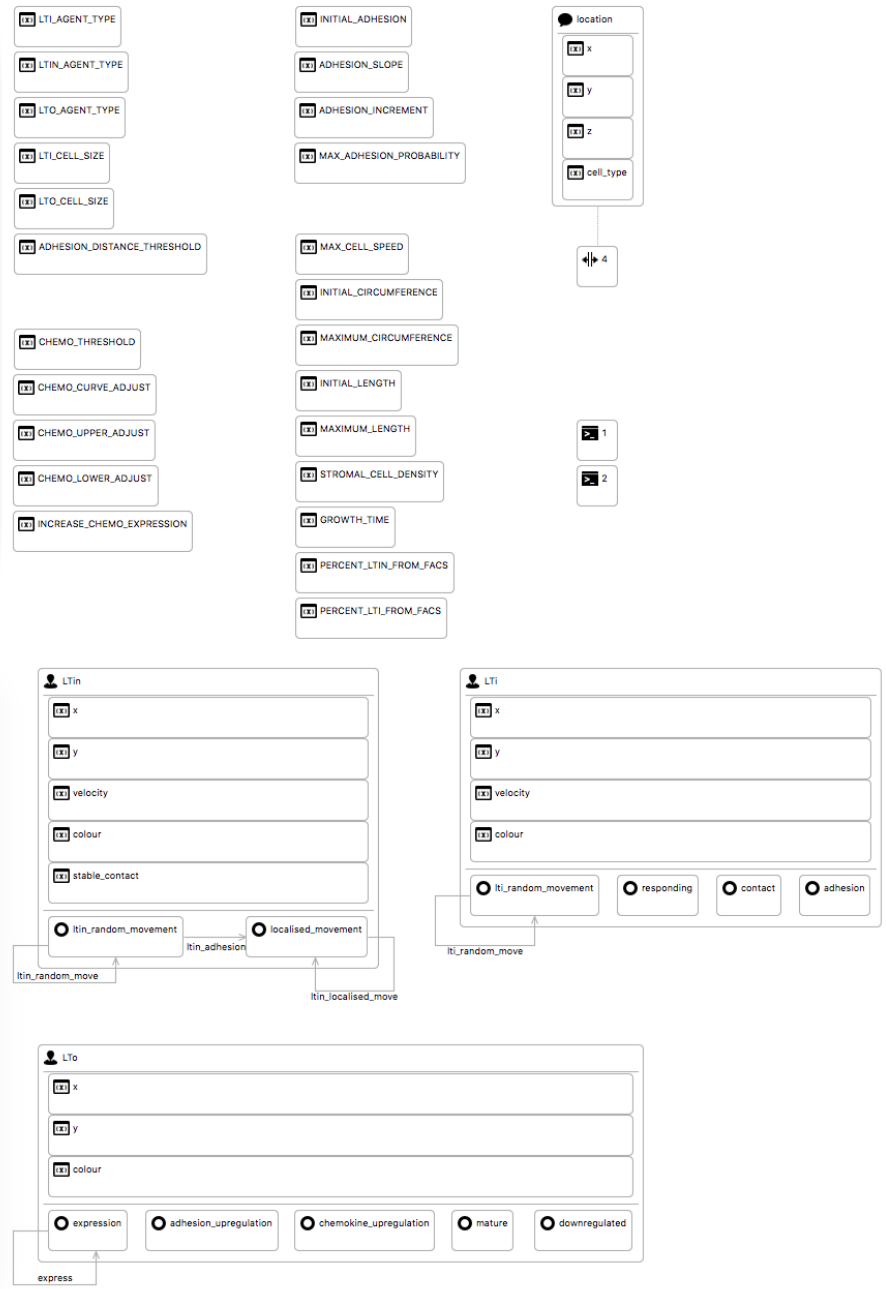
[30] A. Arcuri and L. Briand, '"A Practical Guide for Using Statistical Tests to Assess Randomized Algorithms in Software Engineering"', in *Proceedings of the 33rd International Conference on Software Engineering*, ser. ICSE '11, Waikiki, Honolulu, HI, USA: ACM, 2011, pp. 1–10. DOI: 10.1145/1985793.1985795. [Online]. Available: http://doi.acm.org/10.1145/1985793.1985795.

5 Appendix

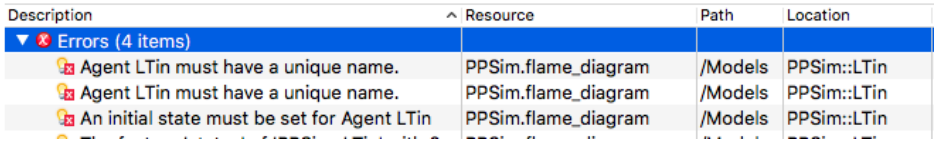
5.1 Simulation Parameters

5.2 Cell Data Structures

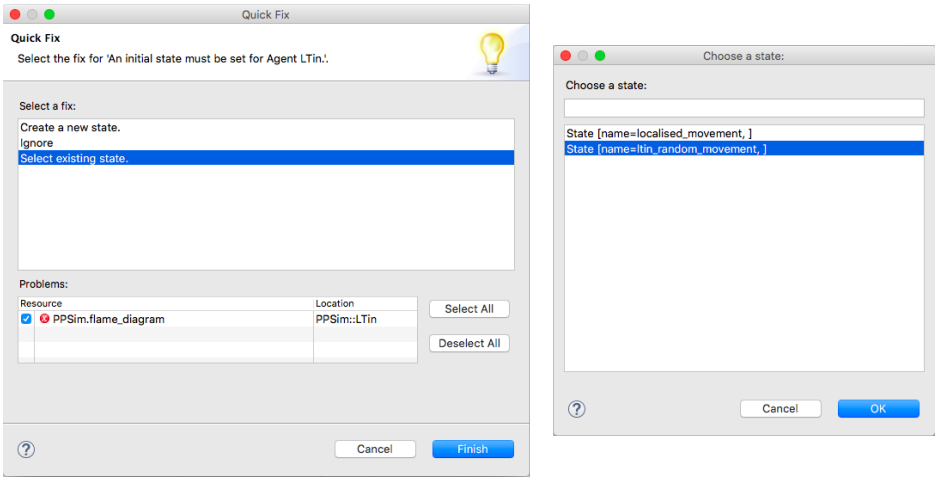
5.3 Other



(a) Graphically produced FLAME GPU Simulation model for Peyer's Patch



(b) Graphical validation using EVL constraints



(c) Quick fix of invalid model

Figure 5.1: Graphical Tool for creating FLAME GPU models