Going around again - modelling standing ovations with a flexible agent-based simulation framework.

Philip Garnett

Department of Anthropology, Durham University, Dawson Building, South Road, Durham, DH1 3LE. UK philip.garnett@durham.ac.uk

Abstract. We describe how we have used the CoSMoS process to transform a computer simulation originally developed for the simulation of plant development for use in modelling aspects of audience behaviour. An existing agent-based simulator is re-factored to simulate a completely different type of agent in 2D space. This is possible and desirable because the original simulator was designed with the intention that it could easily be use to model a variety of different agents interacting in 2D and 3D space. The resulting simulation will be used to simulate the phenomena of standing ovations in audiences as a model system of tipping point behaviour. Continued development of this simulator, assisted by the CoSMoS process, has resulted in a general purpose lightweight simulation framework.

1 Introduction

The dynamics of standing ovations incorporates many interesting aspects of human behaviour and even with a simple computer simulation there are many things to explore. The collective desire of an audience (or at least some parts of an audience) to display their appreciation of something is interfered with by that very human desire to not embarrass oneself. On the face of it this sounds like a difficult system to understand. There are complex individual decisions about how much you want to stand up and clap, or stay firmly rooted to your seat. There is also the behaviour of your immediate neighbours to consider. If the people sitting next to you start standing up, what do you do? Does that overcome your reluctance to get up? Or if you stand up and they don't, what then? Sit down and shrink back into your seat? Then there is the pressure to conform with the wider audience. If everyone on the other side is standing do you spring up to get your side going, or wait to see if the enthusiasm diffuses round to your section? From the perspective of the individual agents (or people) there is a lot of complex decision making going on. However, at the population level could there be a simple set of rules that are governing the global behaviour of the system? In this paper we describe the process of re-factoring an existing simulator that was built to be a flexible agent-based modelling platform, but with a focus on a particular use, to allow us to model standing ovations as an

emergent behaviour. During the re-factoring process the simulation framework has become increasingly modular and generalizable.

A simulation must be developed using a rigorous process of design, implementation, and validation if it is to be scientifically respectable, understandable and reproducible. As we aim to maintain flexibility, the simulation tool will need to be upgraded and enhanced in a principled manner as its requirements change, and we use it to address new research questions. This helps ensure that we fully understand the foundations of the platform before we build something new on top. The CoSMoS (Complex Systems Modelling and Simulation) process [1] provides a flexible approach designed to support the modelling and analysis of complex systems, including the design and validation of appropriate computer simulations.

We have previously used the CoSMoS process to guide the initial development of a simulation of an abstract tissue level model of plant cells [8]. We then used the CoSMoS process to enhance that existing model to produce a more capable and efficient version of the simulation [7]. Here we present work using the same process [1] to guide modification of the basic simulation framework to produce a new simulation of standing ovations in simulated audiences. This work is further evidence that the CoSMoS process can be used in an incremental manner to assist in the reuse of existing simulation code.

In §2.1 we overview the CoSMoS process as used for modelling, designing, and implementing simulations of human behaviour. In §2.2 we discuss the use of UML as a suitable modelling language to support this process. We then use the CoSMoS process components to structure the remaining sections. In §3 we introduce the Research Context. In §4 we summarise the standing ovation Domain Model. In §5 we discuss how the Platform Model was developed using the CoSMoS process. In §6 we conclude with a discussion of our experiences and some preliminary results.

2 Background

2.1 CoSMoS Process: The modelling lifecycle

For this work we use the CoSMoS process as described in detail by Andrews et al. [1], and used in our earlier work [8, 7]. The CoSMoS process provides a systematic approach to building models and simulations of complex systems, and here we apply it to modelling human behaviour. The CoSMoS process is an inherently incremental process without a defined end point. It therefore lends itself to producing a series of simulations aimed at answering a particular question. This flexibility also allows for taking an existing simulation down a new path to answer different questions. We [8, 7] and others [14, 4, 13] have successfully used this process to assist in the production of simulations of complex biological systems. Summarised in figure 1, the version of the process used here contains the following components (summarised from [1], and the description of the process is taken from [7]):

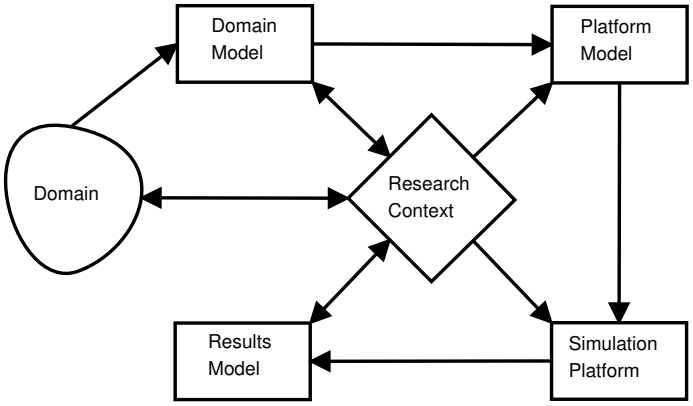


Fig. 1. The components of the CoSMoS process [1, fig.2.1]. Arrows indicate the main information flows during the development of the different components. There is no prescribed route through the process, in so far as going back a step at any point in the process is allowed and often useful.

Research Context : the overall scientific Research Context. This includes the motivation for doing the research, the questions to be addressed, and the requirements for success.

Domain Model : conceptual "top-down" model of the real world system to be simulated. The Domain Model is developed in conjunction with the domain experts, with its scope determined by the Research Context. The model may explicitly include various emergent properties of the system.

Platform Model : a "bottom up" model of how the real world system is to be cast into a simulation. This includes: the system boundary, what parts of the the Domain Model are being simulated; simplifying assumptions or abstractions; assumptions made due to lack of information from the domain experts; removal of emergent properties (properties that should be consequences of the simulation, rather than explicitly implemented in it).

Simulation Platform : the executable implementation. The development of the simulator from the Platform Model is a standard software engineering process.

Results Model : a "top down" conceptual model of the simulated world. This model is compared with the Domain Model in order to test various hypotheses. This part of the process is on-going research.

2.2 Modelling human behaviour and simulations with UML

UML (Unified Modelling Language) [12] is a suite of diagramming notations designed to aid in the development of large object-oriented software engineering projects by groups of developers working in teams. Ordinarily it is used in conjunction with an object-oriented programming language; it has been shown to be well suited to agent-based modelling [11], and the production of agent based

models [7, 14, 8, 6]. Here "agents", representing humans, map naturally to agents that can be described using UML. This allows for much of the structure of the behavioural simulation to be represented in UML. We have also found that UML is suitable for capturing simiplified human mental states that are significant to the model (see §4.1).

3 The Research Context

Although we are modelling standing ovations we are interested in the wider phenomena of tipping points and emergence in human behaviour. We use the standing ovation as a simplified case of tipping points in social systems. Human behaviour on both the individual and population level is very complex. People can frequently behave in unpredictable ways, often doing things that may seem counterintuitive or unexpected. People also have the capacity to consciously go against, or follow the crowd, making predicting an individual's behaviour very difficult. At the population level however human behaviour becomes more predictable. A herd mentality (the desire to belong) may mean that in the short term at least it is possible to predict a future state of a group (or herd) of people [15]. However, it is often the unexpected shifts in population behaviour that we would like to be able to predict or detect. These tipping points are the focus of our wider research and the model presented here was designed as a simplified model of tipping point behaviour.

In the context of wider society social tipping points are very interesting [2, 9, 3]. Even when a human system appears to be relatively stable it could go through a tipping point and change state completely. Here we define a tipping point in a system as when it moves rapidly from one stable state to another state which may or may not be stable. We therefore allow our definition to include reversible changes, so even if the system goes back to its initial state we consider it to have gone through a tipping point. Of particular interest to us are systems that appear to be stable but have the potential to go through a tipping point. In order to understand these systems we must identify the triggers and thresholds that indicate the point at which the system tips.

Modelling an individual human would be extremely difficult but at the population level it might be possible to break the behaviours thought to be of importance in a system down into a set of simple rules. The model can then capture the abstract simple behaviours between agents that when simulated produce the overall systems behaviour as an emergent property. In the model the synchronised human behaviour of standing ovation is an emergent property of the combination of the basic rules of the system, and the interaction between the agents in their environment.

Standing ovations have been used as a model system for synchronised human behaviour in a number of different contexts [16, 10, 5]. We focus on standing ovations as a model system for a social tipping point, paying particular attention to how the system is triggered; what constitutes a trigger; at what point can a

system be considered to have tipped; and does that allow us to predict the outcome of a system?

4 The Domain Model: the standing ovation

Standing ovations capture many interesting aspects of synchronised human behaviour. Individual members of an audience are influenced by how good they thought a performance was, and by the behaviours of the people around. Depending on the relative influence of these factors individuals might jump up and start clapping, with little regard to what others might think, or wait to see if others are going to stand up first. In fact there are many possibilities for both showing appreciation, or dissatisfaction. In §4.1 we outline in more detail the aspects of standing ovations that we are going to capture in the simulation. In §4.2 we summarise how we capture these behaviours in UML.

4.1 The Mental Domain

A standing ovation is an emergent property of the relative influences on individuals in an audience. We break the influences down into three simple parts:

- The individuals own personal enjoyment of the performance. If their enjoyment is very high and they are likely to stand without paying much attention to the rest of the audience. Similarly, if it is very low they are likely to remain seated. The interesting social effects in the system will operate mainly on the people who are somewhere in the middle.
- The behaviour of an individual's immediate neighbours will modify this background likelihood of standing to either suppress it so they remain seated, or activate it causing the individual to stand.
- The final influence on an individual is the room size. We assume that an individual can make an assessment of the larger space and that this acts on their background likelihood of standing in a similar way to the neighbour interactions.

We will also look at what affect an individual being able to stand up again has on the behaviour of the system.

4.2 Domain Model UML

In order to re-factor the simulator for its new use we start the process at the Domain Model UML to get a sense of which parts of the simulator can remain and which need to be wholely removed or significantly altered.

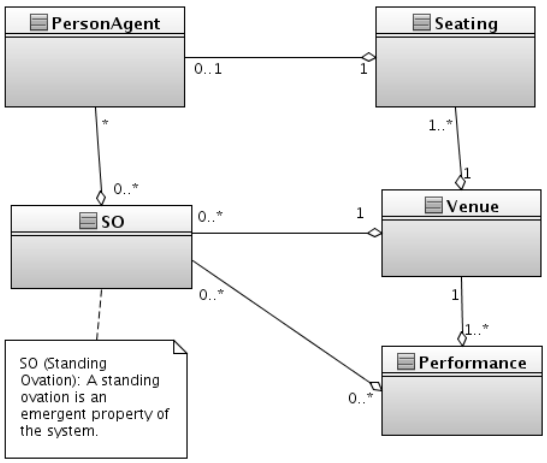


Fig. 2. Domain Model class diagram.

Domain Model use cases. During the development of the biological models use case diagrams were used to capture the higher level interactions that were to be included in the simulation [8]. For re-factoring the simulation for a new use we didn't find it advantageous to revisit the use case diagrams as we have come to the conclusion that all of the relevant interactions can be captured in the description of the domain class and state domain diagrams. Therefore unless it is the case the application of the process would benefit greatly from use case diagrams, they will not be used.

Domain Model class diagram. This captures the different aspects of the system that are required as classes. At this point in the process is it desirable to take forward only the parts of the Domain that we believe to be absolutely necessary. The classes map directly to either the required spatial elements of the system that we need to include in order to simulate standing ovations, or the agents that exist in this space. This includes the Venue, and the Seating within the Venue, which together define the space the system exists in. There is also the class, PersonAgent, which is a simplified representation of the people in the simulation. We also capture the required aspects of the performance in the Performance class. These classes provide enough structure to allow us to model a standing ovation. We also capture the emergent property of a standing ovation SO. At this point it is a desired outcome of the interaction between the Venue, Performance and PersonAgent classes (See figure 2).

Domain Model state diagrams. When describing a biological process state diagrams prove very useful as they can easily show the different states that biological objects can exist in and how they move between states [7]. In figure 3 we can see that in this model the number of physical states the PersonAgents

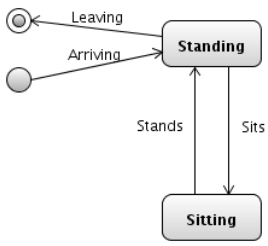


Fig. 3. Domain Model PersonAgent state diagram.

can be in is very limited, Sitting or Standing. We therefore propose that in the case of modelling human behaviour (or any behaviour that is not manifested in a physical change) it is advantageous to also model more abstract mental states that are of importance to the model. These can then be included in the diagrams which assists communication of important parts of the simulation. In the mental state diagram, mental states provide information about what needs to be captured by the model in order for the PersonAgents to transition from the Sitting to the Standing state. Figure 4 shows the mental state diagram for a PersonAgent. When a PersonAgent changes state they can go through a loop of mental state transitions before they commit to either remaining in the same state or transitioning to a different state. This loop incorporates what the individual is considering doing, the state that their neighbours are in, and the different states that the wider audience is in. In figure 3 it is possible to clearly label the state transitions as Stands and Sits. However, when the mental processes are included the transitions are more difficult to label as the physical state might not change even though the PersonAgent has gone through a process of assessing whether it is going to change state.

In some sense what we are attempting to capture by including mental states in the Domain Model state digrams is an agents abstract individual tipping point. The point at which an agent is required to make the decision to change state they (for a short time) are no longer simply passively Sitting or Standing, but instead are in a separate state of making a mental decision. The outcome of this rather abstract mental state is either the Sitting or Standing, but during the process a transition has occurred.

Upon entering the venue people are standing. A mental process then occurs during which individuals assess what they want to do, taking into account the state of the wider audience and their neighbours, in order to find their seat. Throughout the performance this process will continue as there are a number of possible reasons why a person may have to transition from a Sitting to a Standing state. For the purposes of this simulation we are only interested in people's behaviour at the end of the performance.

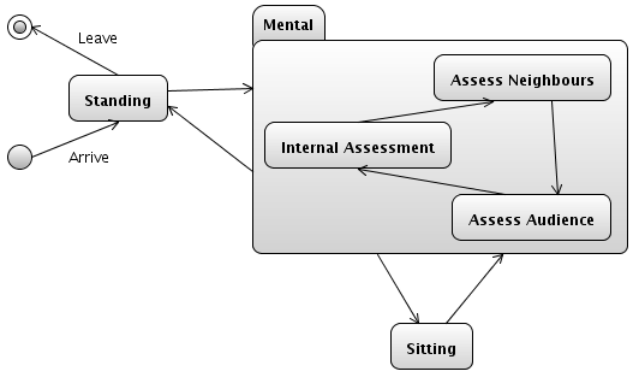


Fig. 4. Domain Model mental PersonAgent state diagram.

5 The Platform Model

The Platform Model includes all the extra components that allow the simulation to run. This includes all the processes required to get the simulation to a point where it is able to start, and the components identified in the Domain Model that are important to allow the behaviours of interest to occur. Also included are any additional behaviours that are required for the model to function but that might not be of explicit interest to the Research Context. These can be implemented with a view to producing an efficient simulation rather than system fidelity. Finally, we need to include aspects of a simulation that are not part of the Domain but are required in order that simulation results may be observed and documented.

5.1 Platform Model UML

The platform model UML bridges the gap between the Domain Model UML and the final implementation of the model in code.

The Platform Model class diagram. The is produced from the Domain Class diagram, with all emergent properties (such as a standing ovation, SO) removed. Shown in figure 5 this is a high level diagram and includes the four main components of the system, the Venue, PersonAgent, Seating, and Performance. There is only one Venue, we assume that it has at least one Seating. That Seating can either be unoccupied or have one PersonAgent in it. A Venue is assumed to have at least one Performance.

The Platform Model class diagram, implementation level. This diagram represents the structure of the underlying code of the simulation by including implementation level data structures and any generalisable classes. Figure 6 shows

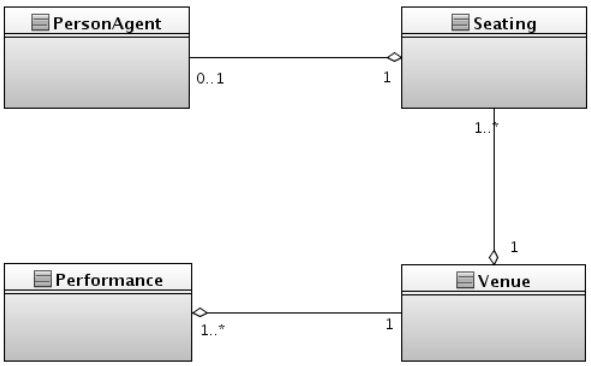


Fig. 5. Platform Model class diagram: included are the four main components of the system being models, the Venue, PersonAgent, Seating and Performance.

the implementation of the classes carried forward from the Domain model. Seating is a child of the Area class. In this simulation each Area can only hold 0 or 1 PersonAgents, which is implemented as a specific class. The Areas are stored in the Space, and to improve the performance of the simulation can either be (in conjunction with a suitable programming language) separate threads of execution or grouped together in the separate executable buckets of Areas (executable buckets are used in the Java implementation but left out of figure 6 to improve readability). Structuring the Space in this way has advantages for investigating the affect of the interaction between agents and the spacial environment, and allows easy reproduction of interesting spaces (see §5.3 and §6).

Space has at least one SeatingZones, and a SeatingZones has at least one Area. There are also EmptyAreas which are used for Space that are not processed by the running simulation. The Space, Performance and SeatingZones are stored in the Venue class. The purpose of the PersonAgent class is to contain the current state of the PersonAgent. The PersonAgent's interaction with the wider system is mediated via the Seating and SeatingZones classes. In its current implementation the Performance is almost unnecessary but in the future when we extend the simulation to include more interaction between the performance and the audience the complexity and importance of this class will increase. The Venue class acts as a container for the simulated system. It could also fulfill any more abstract requirements that the venue might have.

5.2 Comparing two models

The inclusion, during redevelopment, of a flexible Space makes it easier adapt the simulation to its new purpose. The original modelling framework had at its centre a flexible and extendible class that is the basis for all the different types of space in the system [7]. This flexibility allows for the many different types of space to be described. In this simulation the Area class that is held in the

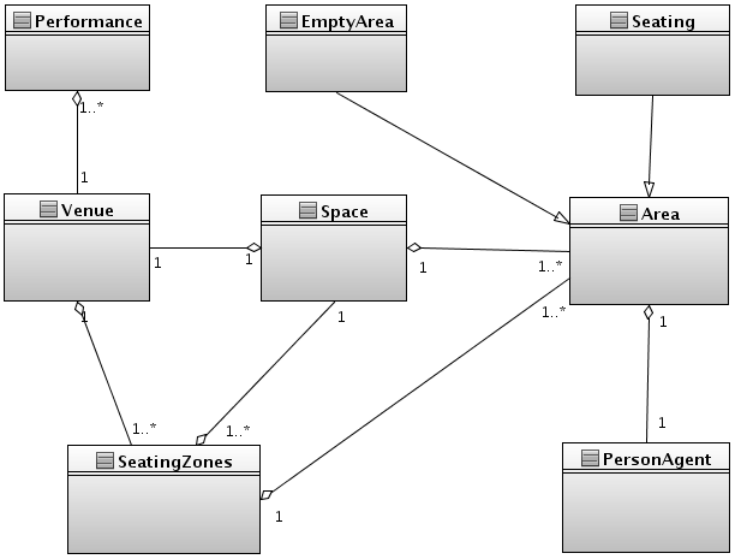


Fig. 6. Implementation level Platform Model class diagram: The Venue has one Space, at least one Performance, and at least one SeatingZones. The Space and is composed of many Areas which can be one of the two child classes EmptyArea or Seating. Seating belongs to one SeatingZones.

Space class is extended to describe seating (child class Seating). Each area of Seating contains within it one agent representing a single person, PersonAgent. The Seating class connects that agent with this environment. Via the Seating class the agent can contact its neighbours, it can also sample from all of the Seating areas of the simulation to gather information about any of the agents in the system. Figure 7 shows the Platform Model class diagram for the auxin simulation model described in [7]. By comparing this diagram with figure 6 is it possible to see how at the implementation level very little of the underlying structure of the model needed to be altered for its new purpose.

The behaviour of the model is controlled by running a method within the extended Area classes called 'process'. This processes any agents held in this part of the space. What the different agents do is controlled by their implementation. This structure means that no assumptions about how the agents are behaving in space are transferred from one model implementation to another. What does remain the same is the channels through which the agents get information about their environment. However because this is ultimately determined by the extended Area classes the exact information that flows is also specific to each model. To move from an auxin transport model to a standing ovation model the Area class was extended to describe the Seating in the Venue. The auxin model's Plant class, which is a general holding class, providing functionality similar to a database, was simply renamed as Venue. The Cell class in the auxin model

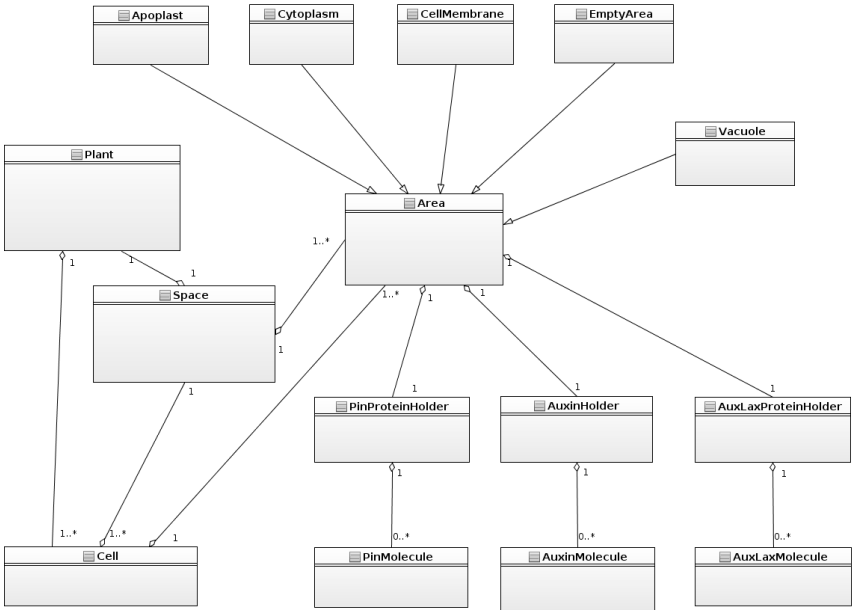


Fig. 7. Implementation level Platform Model class diagram for the original auxin simulation:

implemented an abstract collection of different Areas (three different types, Cytoplasm, Membrane, and Vacuole, see figure 7). For the standing ovation model this class is completely reimplemented as SeatingZones. SeatingZones is equivalent to Cell as it is a collection of areas of Seating that can be considered as 'together' in the space, and information may flow differently between different zones or within a zone. Currently these spatial zones are not generalised from a parent class and therefore for this model the class was rewritten from the ground up to ensure it provided the correct functionality.

During the process of reimplementation it became clear that the Cell and SeatingZones classes share a number of common features. We are now working on two generalizable classes, SpatialNetwork and AgentNetwork. SpatialNetwork provides a basis for the implementation of abstract collections of spatial Areas, which Cells and SeatingZones are examples off. AgentNetwork provides a basis for abstract collections of agents. In the case of our standing ovation model this could be men and women PersonAgents for example. These should not be confused with the executable buckets of Areas mentioned in §5.1. It would however be useful to implement the multi-threaded execution of the simulation with these abstract collections if that seems feasible and desirable. The generalizable form of the future framework is shown in figure 8. The introduction of these higher level collections brings this framework closer to the framework described in [4, 13] (which the author was involved in developing). The modular design is deliberate as it helps to make clear what the foundations of the simulator are, these

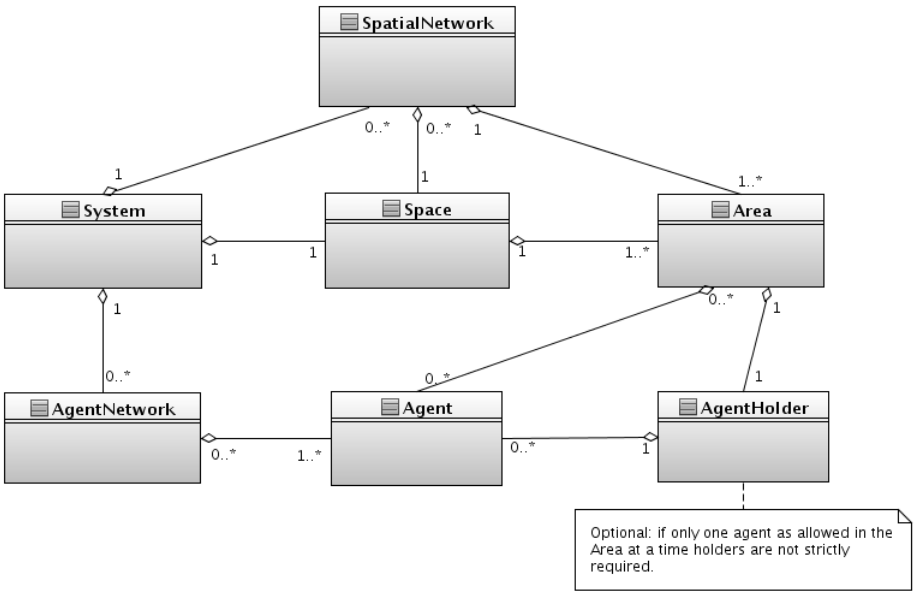


Fig. 8. The class diagram of the generalisable simulation framework.

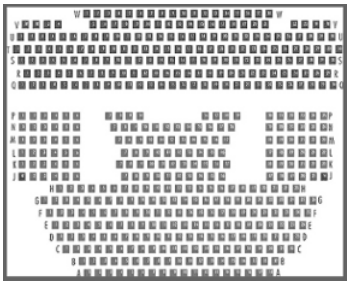
foundations can be extended helping avoid the reusing code implemented for old models that are not suitable for the new purpose. The modeller is encouraged to start from the generalizable classes not a previous implementation.

Platform Model state diagrams. These remain the same as the Domain Model State Diagrams.

5.3 Space from Templates

The original auxin model was designed to allow for cell tissues to be loaded into the simulation from a section through a real tissue [7]. This feature allows the standing ovation model to load in approximations of real seating plans. These seating layouts are read from template images into the simulation when it starts. The template provides information on the position of all the seats in the system, and how they are arranged in 2D space. Including any empty areas or gaps between different sections of the audience.

This allows us in future to explore the effect of boundary conditions in the system. Not only those at the edge of the seating, but also boundaries within the seating. It is possible that the probability of a standing ovation fully taking hold might be affected by the seating in many venues being in sections. The small breaks in the seating may alter the dynamics of system, and the flow of information within the system, changing how the audience responds as a whole. It might be that the sectioning of the audience encourages a greater degree of



(a) A real seating plan of a lecture theatre.

(b) A template approximated from the real plan. For this simulation we chose not to use seating zones but they could have been used to produce a more faithful plan.

(c) A still from the running simulation. The light grey area of the seating indicates that the PersonAgent is standing.

Fig. 9. By allowing the use of real seating plans we can compare real data with simulations allowing the investigation of effect of changing the layout of the venue on the system.

autonomy (I'm part of this group, the people overthere are a different group), reducing the ability of a standing ovation to spread over the entire system, or perhaps giving small sections of the audience the confidence to act alone.

6 Discussion

We have been able to show that the CoSMoS process can be used to assist with the implementation of an existing model in a new area of scientific study. This is in some ways an inherently dangerous method of producing a simulation. There is the potential to use code that when originally written had at its foundation assumptions that are not suitable for the new purpose. Where the CoSMoS process becomes valuable is that it insists that the model implementation starts from a suitable place: the Domain. Starting at the domain and working forward provides useful information about which parts of the original model can be safely reused. Once the unsuitable parts of the system are removed the process can then be followed as normal to produce the new simulation.

This systematic process has two added benefits. Simply casting a simulation into UML along with using the CoSMoS process can often highlight possible improvements that can be made to the model and resulting simulator. Good examples include the identification of code that has been put in the wrong place, and the discovery that parts of the system that have been excluded from the Domain Model or not explicitly identified when they need to be. One example is that there is a temptation to include aspects of agent behaviour in both the Area and Agent classes which could be a source of confusion further on in the development process. Avoiding this potential confusion also increases the modularity of the simulation by encouraging consistency in the placement of methods. Increasing the modularity of the simulator has helped in the development of an efficient generalizable multi-threaded agent-based simulation framework. In fact the generalizable framework emerged from the process. The adherence to the CoSMoS approach has also ensured that its development has been systematic and well understood. At each stage of the process effort has been made to understand and acknowledge the decisions that have been made and why, and many of the decisions are recorded.

It is true that the simulation framework produced is not completely generalizable and could not be used to produce *any* agent-based simulation. It is unlikely that any such tool could be produced. We are also of the opinion that the attempt should not be made. A completely generalizable framework would either be susceptible to bloat in both size and complexity, resulting in a tool that was difficult to maintain, fully understand, or apply appropriately. Or it would become such light-weight collection of extendible classes that it would be of greater advantage to develop a system of patterns instead. There then remains the question of how do you determine when a general tool should be used over a one-off efficient simulator? This is not an easy question as this framework started as an efficient one-off simulator and has developed into a more generalizable tool. The CoSMoS process has a lot of offer as a way of assisting this decision making process. Even though we intuitively believed that the foundations of simulation of auxin transport were suitable for modelling another 2D agent-based system, following the CoSMoS process assisted in determining which parts could remain and which needed to be rewritten or simply removed. The CoSMoS process should therefore allow the developers and domain experts to determine if there

is a suitable existing tool (as long as they actually understand what the tool has to offer), or if a new simulator is required.

Acknowledgements

We gratefully acknowledge the financial support from the Leverhulme Trust who funds the Tipping Point project based in the Institute of Hazard, Risk and Resilience at Durham University. We would also like to thank the developers of the CoSMoS process. Finally we would like to thank the reviewers for their detailed and very helpful comments, and Lauren Shipley for her assistance with proof reading.

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, University of York, 2010.
2. M Batty. Discontinuities, tipping points, and singularities: the quest for a new social dynamics. *Environment and Planning B: Planning and Design*, 35(3):379–380, 2008.
3. William A Brock. *Tipping Points , Abrupt Opinion Changes , and Punctuated Policy Change by*. PhD thesis, University of Wisconsin, 2004.
4. Alastair Droop, Philip Garnett, Fiona A C Polack, and Susan Stepney. Multiple model simulation: modelling 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*, pages 79–111. Luniver Press, 2011.
5. Fernando Eesponda, Matías Vera-Cruz, Jorge Tarrasó, and Marco Morales. The complexity of partition tasks. *Complexity*, 16(1):56–64, 2010.
6. S. Efroni, D. Harel, and I. R. Cohen. Towards rigorour comprehension of biological complexity: modeling, execution, and visualization of thymic T-cell maturation. *Genome Res*, 13(11):2485–2497, 2003.
7. Philip Garnett, Susan Stepney, Francesca Day, and Ottoline Leyser. Using the CoSMoS Process to Enhance an Executable Model of Auxin Transport Canalisation. In S Stepney, P Welch, P. S. Andrews, and A. T Sampson, editors, *CoSMoS 2010*, pages 9–32, 2010.
8. Philip Garnett, Susan Stepney, and Ottoline Leyser. Towards an Executable Model of Auxin Transport Canalisation. In W P Stepney Susan, Polack Fiona, editor, *CoSMoS 2008*, pages 63–91. Luniver Press, 2008.
9. Suzanne B. Goldberg. Constitutional tipping points: Civil rights, social change, and fact-based adjudication. *COLUMBIA LAW REVIEW*, 106(8):1955–2022, 2006.
10. John H Miller and Scott E Page. The standing ovation problem. *Complexity*, 9(5):8–16, 2004.
11. J Odell, H Parunak, and B Bauer. Extending UML for agents. In *AOIS Workshop at AAAI*, pages 3–17, Austin, 2000.
12. OMG. Maintainer of the UML Standards., 2012.
13. 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*, pages 113–133. Luniver Press, 2011.
14. Mark Read, Jon Timmis, Paul S. Andrews, and Kumar Vipin. Using UML to Model EAE and its Regulatory Network. In Paul S. Andrews, Jon Timmis, Nick D. L. Owens, Uwe Aickelin, Emma Hart, Andrew Hone, and Andy M. Tyrrell, editors, *Proceedings of 8th International Conference on AIS*, volume 5666 of *Lecture Notes in Computer Science*, pages 4–6–6, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
15. E.M. Rogers. *Diffusion of Innovations*. Free Press, New York, 1962.
16. Miklos N Szilagyi and Matthew D Jallo. Standing ovation: an attempt to simulate human personalities. *Systems Research & Behavioral Science*, 23(6):825–838, 2006.