Socio-Physical Modelling and Declining Mobility

Penny Faulkner Rainford[1,3] and Susan Stepney[2,3] and Yasmin Merali[1,3]
[1]Center for Systems Studies, University of Hull, UK
[2]Department of Computer Science, University of York, UK
[3]York Cross-disciplinary Centre for Systems Analysis
p.rainford@hull.ac.uk

Abstract

We present a new modelling approach for complex systems incorporating a dynamic environment and individuals with agency. We do this through multiple models at different levels. We develop a common meta-model for these kinds of models. The meta-model captures the concepts of agents moving and interacting on a dynamic network, to provide the power of an agent based model situated in the context of a dynamic and changing environment. The addition of context allows us to isolate the decision process of the agent from the constraints and resources provided by the environment, so we can consider separately the effect of changes in the environment from changes in the agents' decision process, and changes caused by agents acting differently based on their learning from, and adapting to, the changed environment.

We develop a generalised platform model for implementing different complex systems conforming to the meta-model. We illustrate the approach by developing a domain model for a particular system of interest, a simplified model of declining mobility, which we use to guide the specialisation of the generic platform model to an implementation and to perform simulation experiments.

Introduction

Agent based simulation has long been a central method in modelling for social sciences (Gilbert and Terna, 2000; Bonabeau, 2002; Epstein, 2006; Macal and North, 2010). For a field in which people, their interactions, communications and choices, are the main interest agent based simulations are a natural choice. Modelling people as individuals is important as part of studying the systems and societies we build. There are several features of complex learning agents in a complex environment that need to be considered when modelling.

Typically, a social system is complex, rarely truly isolated from others, and not under the complete control of the individuals within it. There are external factors and decision makers. Within the system there are options that some individuals are not able to avail themselves of; there are limitations to their knowledge and perception of the system, which in turn limits their choices. In a pure agent based model, these constraints have to be captured within the agent model and implementation.

A social system comprises unique individuals who have different interactions with the system dependent on their own circumstances. Individuals can learn, and their behaviours can change over time. Their learning is shaped by the consequences of their choices, and their choices by the results of their learning.

Such complex sets of choices and decisions, constraints on those choices, and learning are common aspects of existing agent based models. Some of them also have heterogeneous agents who have different capabilities and choices available at different times. This can be modelled through internal agent state or different agent types. This allows individual agents with different behaviours, and a scope for changing state, behaviours, and types over time.

Constraints placed on agent choices have been modelled either using limited communication networks (Blythe and Tregubov, 2019), or using spatial grids with different resources (Rahman et al., 2007; Henscheid et al., 2006), or regional and class grouping (Palmer et al., 2015). These provide a form of environment for the agents. The location of the agent or their ability to communicate with other agents would determine their position in the environment. There are systems that use databasing to centralise information shared between agents, a form of environment, such as the FARM system (Blythe and Tregubov, 2019), which does this to enhance parallelisation.

Social systems do not have static environments; systems and services change over time and react to individuals' responses, general demand, and policy changes (Silverman, 2018). So the modelled environment should be able to be dynamic in response to actions of individual agents, to behaviours of the overall population of agents, and to external events. Modifying a spatial grid by moving resources, or changing the properties of an agent and its communication links, can give a limited dynamic environment.

However, these forms of dynamic environment are limited to affecting individual agents or based on external timing for events. These environments lack reflexivity: they do not react to themselves separate to the agents or to the behaviour of the population as a whole. Although they can capture

some of the complexity of a dynamic environment, they are not necessarily the most natural representation to use with a domain expert to capture the essential features of the model. Design choices may be lost in translation between the intent of the domain expert and the detailed code produced by the simulation programmer without the ability to discuss a high level system design back and forth.

Here we use a dynamic network with its own state and behaviour for our environment model. This provides a sufficiently powerful representation of many kinds of complex dynamic environments, able to provide input for agents' decision making. A network environment model provides natural limitations on an agent's options, based on limitation of information and choices available at the current location, and limitations on movement in the system through the edges and nodes of the network. Also, a network representation is a natural model for communication between a domain expert and simulation programmer.

A dynamic network allows these agent constraints to change over time. Depending on what system is being modelled, agents may modify the network directly, or the network may change itself using intervenors in response to larger patterns in agent activity or to external events (Merali, 2006).

We have created NetSim, a modelling framework for socio-physical modelling, which extends plain agent based modelling with a complex and dynamic environment that can facilitate joint development of the model with a domain expert. NetSim provides agents with a context, so they can be defined separately from the environment. It also provides a complex environment with spatiality and structure that can change over time.

We can use NetSim to study agents' reactions to change. Because we are using a dynamic network, we can readily modify the variables of our network through intervenors and change its structure, with confidence of the independence of our changes. NetSim should allow for modelling of more complicated systems, carefully separating out the agent properties, the environmental properties, and their interactions. This should make it simpler to model agent and environmental behaviours, and to explore the consequences of changing them independently.

Rationale

The importance of environment on systems has been shown in many different contexts. It has repeatedly been shown that system models respond differently in different environments. An example from ecology is the theory of island biogeography (MacArthur and Wilson, 2001), which posits that distance from the mainland and size of island are as significant in the extinctions caused by invasive species as are the species themselves. In plant growth modelling, environmental L-systems (Měch and Prusinkiewicz, 1996) incorporate a model of the environment alongside models of the growing

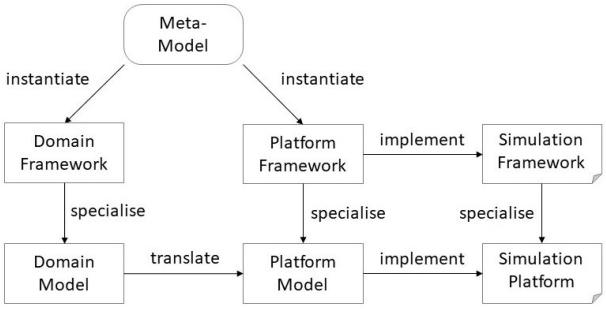

Figure 1: The relationships between the meta-model, the models, and the code. The domain framework model is an instantiation of the meta-model in domain terms suitable for communication with the domain expert; the domain model is a specialisation of this for a specific problem. The platform framework is an instantiation of the meta-model in generic implementation terms capturing generic agents and networks. The platform model is a specialisation of the generic platform framework, specialised with (suitably translated) domain components. The simulation framework and simulation platform are executable implementations of these models.

plants, and clearly demonstrate the effect of different environmental conditions such as regions with varying illumination. In animal behaviour (Győri et al., 2010) the context of the situation changes the animal's reaction to others.

As complexity increases it becomes harder to define the boundaries of a system. In addition to defining the objects of interest, it is important also to find a way to include the surrounding interacting elements. This can be done by incorporating the salient environmental features into the complex system model.

We can also use the environment as the means to mediate communication between agents, by using the technique of Environment Orientation (Hoverd and Stepney, 2015), which results in all communication being local. This can reduce the complexity of multiple agent-to-agent communications. We use this as the concept behind separating our agents and environment in our model.

Model and meta-model

Here we follow the CoSMoS (Complex Systems Modelling and Simulation) approach (Stepney et al., 2018) by developing not just an implementation model but three separate kinds of models (figure 1). First, the *meta-model* defines the concepts used for this type of modelling. Then we have two specific models for any particular implementation of the system. The *domain model* uses the concepts of the meta-model to define the problem domain, with the assistance of domain experts. This model defines the system in a way that matches the domain experts' understanding, while placing clear boundaries on the system to be modelled. Finally, the *platform model* is the model that is to be implemented in software. It also uses concepts from the meta-model.

Here we use two levels of models. We define generic

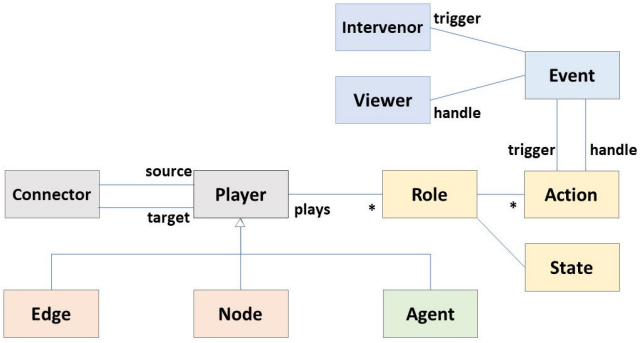

Figure 2: The components of the socio-physical metamodel, which combines agents and networks. The modelling language used is UML; colour is used to highlight different parts of the meta-model. Player is a generalisation of Edge, Node, and Agent. A Connector associates Players; there are three subclasses of Connector (not shown): link connectors associate Nodes and Edges into networks; location connectors associate Agents with the Nodes where they are located; population connectors associate Agents into (sub)populations. Role associates Players with the State and Actions that define their behaviours. Events associate Actions, Intervenors and Viewers: Actions and Intervenors can be triggers of events; Actions and Viewers can handle (receive and respond to) Events.

frameworks suitable for NetSim domain and platform modelling. Then we specialise these, to create a particular domain model and its corresponding platform model. The code for the simulation framework is available: github.com/faulknerrainford/SPmodelling

Meta-model

There are a variety of agent based meta-models developed for different purposes in the literature, for example, Omicini et al. (2008); Klügl and Davidsson (2013); Purvis et al. (2014); Banzhaf et al. (2016). Our NetSim meta-model captures the concepts instantiated to build our socio-physical models, combining concepts from ABM and from networks, see Figure 2.

Whether to model an entity as an Agent or a network Node is a *modelling decision*. Agents would tend to be active entities that 'decide' and 'chose' – to have *agency* – whereas Nodes would tend to be more reactive entities, passively following the rules of the world. However, this is not necessarily always the case: for example, a simple social network model might dispense with Agents, and use Nodes to model the social entities; a physics-based ABM might have purely reactive agents (such as molecules) flowing through a spatial network, according to the laws of physics.

Network: Node, Edge. We use a network to model an environment. A system may have more than one environment of interest, hence more than one network (for example, a spatial network and a social network). A Network is an association (a graph) of nodes and edges.

- **Node** : This models a place in the environment; agents are located at nodes. A node has its own state and behaviour, modelled as a Role.
- **Edge** : This models a route between places in the environment; agents can move between nodes only along edges. An edge has its own state and behaviour, modelled as a Role.

Agent, Population

- **Agent** : This models the actors in the system. An agent is located at a node a network; an agent may be located in multiple networks. The node provides the agent's local environment. An agent can play many Roles.
- **Population** : an association of Agents that are considered worthy of a group identity, for example, all agents playing some role, or at some location, or in some network; membership may change over time.

Role, State, Action, Event

- **Player** : a generalisation of Agent, Node, and Edge, allowing all of which to play one or more Roles
- **Role** : Role associates Players with Actions and State; it models a behaviour of players. It allows different categories of behaviour, eg, student/teacher, farmer/policy maker/customer. All players playing the same role have the same behaviour specification; the behaviour instance is modulated by that agent's current state in that role – eg a customer may be more or less risk averse. Roles have state and perform actions. A Player may play multiple Roles, over time, or simultaneously – eg, a person may be a student on one course whilst teaching another. In particular, agents may play different roles in different networks.
- **State** : the model of the local state of a Player in its respective Role, which it can use to decide actions, and which can change as a result of acting. State change allows learning.
  - an agent can have private state, that only it knows, and public state, which is visible to its node, for the node to provide to other enquiring agents (through Environment Orientation). An agent can also have 'extrinsic' state: state it is unable to make private, for example, its location (a node has to know which agents are located at it).
  - an agent can see its node's state, and the state of an edge it traverses. It cannot directly see the state of another agent, or another node: it must request this information through its node (Environment Orientation)
  - a node can see the state of its edges, and the state of the nodes at the other end of its edges.
  - an edge can see the state of the nodes at its ends.
- **Action** : the model of the individual actions that players can perform. This model may be instantiated as a state

transition diagram, or some other formulation of state change.

- **Event** : Event associates Actions with each other, and with Intervenors/Viewers, providing communication between Players, and with the user. Actions and Intervenors can trigger Events; Actions and Viewers can handle Events.

System The system comprises all the players. This metamodel is 'flat': all agents and network nodes are at the same level. It is simple to extend this to a hierarchical model, where network nodes can be associated with not only with the individual agents located there, but also (sub)systems of players (networks and populations), by use of the Composite pattern (Gamma et al., 1995).

Dynamics concepts govern how things change (not shown in the figure).

- state change : players have mutable state, which they can change with their actions. Agents additionally can change the state of nodes and edges, but not of other agents (this is part of the underlying Environment Orientation of the model)
- lifecycle : players are created/born, they change/behave/act, and finally they are deleted/die
- movement : agents can move between the nodes of their network, along network edges

Intervenors, Viewers are used to provide an interface between users and the system, to allow experiments to be performed, based on the "Model-View-Conroller" (MVC) pattern (Krasner and Pope, 1988).

- **Viewer** : observing (parts of) the system over time, via handling certain Events
- **Intervenor** : making changes to the system during its lifetime, by triggering appropriate Events. An Initialiser is an Intervenor used to specify the initial state of the system.

Demonstration case study

Here we describe a small example to demonstrate the use of the NetSim socio-phyical model framework. This covers the description of the platform model, a specialisation of the platform framework.

We use the example of the effect of falls and deteriorating mobility in the elderly. This particular problem has factors such as the limitation of access to resources over time due to the difficulty of reaching them, and that in different environments an individual's mobility can be affected differently; for example: sitting in a hospital bed reduces the chance of a fall but causes further loss of mobility; occupational therapy might improve the agent's mobility, but can be hard to get to if mobility is poor.

In this example context and environment is crucial to agent behaviour and options. Separating the choice the agent makes from the limitation on options made available to the agent by the medical and social care services allows us to consider the effects on the agents of changes to their environment without changing the agent model.

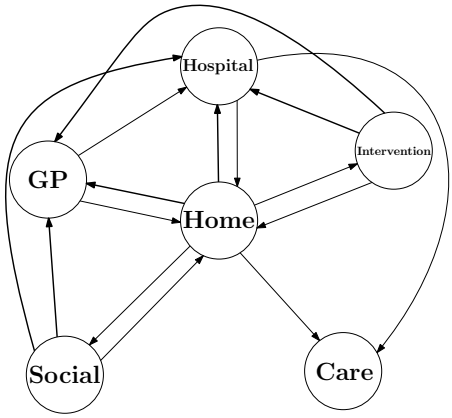

Figure 3: Mobility network, structure showing nodes and edges for case study

This case study uses only some of the capability of our framework; more extensive models are being developed. This is a small initial example of the instantiation of metamodel and specialisation to model a particular domain and problem. It does not use multiple agent roles or types, or multiple networks; the single network has a static structure during an experimental run. However, the parameters of the model are dynamic and controlled by an intervenor, based on information from a viewer.

We focus here on the platform model, as relevant to the work presented here; for the full simulation code implementation see github.com/faulknerrainford/FallModel, version 0.0.1, which specialises the generic simulation framework.

Network

Our model captures treatment of declining mobility through six key services, each modelled as a node in the mobility network (figure 3). The Home and Social nodes capture the 'normal' services for agents; these represent individuals staying at home with minimal movement, or engaging in social activities outside the home involving more movement and interaction with others. Then we have the medical services: Hospital, GP and out-patient Intervention. In response to events an agent might go to the Hospital for treatment, or to a GP for advice. They might also be referred to an out-patient Intervention in the form or physio- or occupational therapy. Finally we have the state of having become unable to continue living independently in their home; the agent is in Care.

There are several edges in the network. There are edges from the Home node to all other nodes. The Home node is a base for the agents; the agent can go to any of the other nodes from there. There are return edges from all these nodes to

Algorithm 1 Perceived choices after Fall

m := Agent.Mobility
if $m < 0$ **then**
    return Care
else
    r := random(0:1)
    if $r < e^{-3m}$ **then** ▷ Severe fall
        return Hospital
    else if $r < e^{-2.7m}$ **then** ▷ Moderate fall
        return GP
    else if $r < e^{2.1m}$ **then** ▷ Mild fall
        return [Intervention, Social]

Algorithm 2 Node states

Home
    Queue: Dictionary {Time:Agent List}
Hospital
    Queue: Dictionary {Time:Agent List}
Intervention
    Capacity: Integer
    Load: Integer

the Home node, except from Care, which is a sink node.

Agents can fall at any node except the Hospital or the GP. The Hospital has no falls as patients in hospital are assumed to be in bed or monitored as they move such that any falls have no lasting effect. The GP is a location an agent attends for a short period of time and is mostly spent sitting, likely with more care since these agents go to the GP only after a fall. The falls come in three varieties – severe, moderate and mild – and occur at random based on the agent's mobility (Algorithm 1). The severe falls direct agents to the Hospital, moderate falls direct to the GP; a patient might not go to Hospital for a fall but still be concerned enough to go to see their GP.

There are also edges from GP to Hospital and Hospital to Care. If an agent's mobility is particularly low when they go to see the GP they are redirected to the Hospital. The edge from the Hospital to Care exists as agents with very low mobility are not sent Home but directly to Care.

Where it is possible for an agent to spend more than a single time step at a node, the length of time an agent will be there is predicted on arrival and is used to add the agent to the node's queue, a list of agents to be processed at that time step (Algorithm 2). To process a time step we work through each node. If a node has a queue type, only the first entry in the queue is evaluated. If a node does not have a queue type all the agents at the node are evaluated.

Nodes

Nodes provide context and environment to agents. When an agent attempts to make a decision about where to move next, it takes its initial perception of options available from the node. The node assesses if the agent falls (as this is more environmental and not in the control of the agent) or if other

Node	Energy	Mobility
Home	$0.3\,t$	$-0.015\,t$
Hospital	$0.2\,t - 0.8$	$-0.1\,t - 0.25$
GP	-0.3	-0.1
Social	-0.4	0.05
Intervention	-0.8	0.3

Table 1: Parameter modification by nodes where t is the integer timesteps the agent spends at the node, includes edge costs

Algorithm 3 Agent State

Agent
    Energy: Real
    Mobility: Real
    Well-being: Healthy | Fallen | At Risk
    Referral: True | False

Algorithm 4 Capacity limitation

c := Intervention.Capacity
l := Intervention.Load
if $l \geq c$ **then**
    remove Intervention from PerceivedChoices

nodes are at capacity. In the case of a fall a deterministic option is presented to the agent of seeking medical help, either in the form of a hospital or GP visit. Assessing both the fall and response uses the state of the agent but is determined by the environment. After fall assessment then the home node performs an additional check based on the capacity of the Intervention node.

The nodes affect agent mobility and energy (Table 1). The Home and Hospital nodes cause an agent's mobility to decline per time step they are located there. The Social and Intervention nodes host agents for only a single time step, and both increase mobility. The Intervention is more effective than the Social node, but requires twice the agent energy. Energy is replenished in Hospital and at Home. Energy in this case is a simplified view of the need for the agent to have personal resources; in an expanded system it might be suitable to consider physical energy alongside some form of emotional resource such as motivation. The GP node does not affect mobility or energy itself, but redirects the agent and provides referrals to Interventions, which have energy costs and mobility requirements. Edges to the GP and Hospital nodes representing falls also cause large drops in mobility and energy (included in the values in Table 1), depending on their severity. Agents are also informed by the environment if they have a 'mild fall', which affects their mobility but does not prompt medical intervention.

Agents

Agents (Algorithm 3) move in the network, based on the filtered options shown in figure 4. These are filtered first by the node, which removes edge choices, based on agent state,

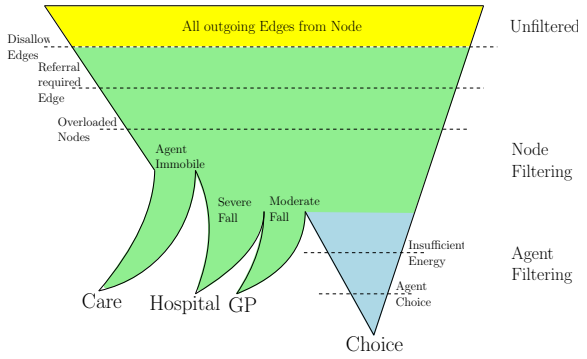

Figure 4: Filtering of option from outgoing edges of a node in the network. These are filtered by the node and then the agent.

Algorithm 5 Referral system

```
if Agent leaving Intervention then
    Agent.Referral := (Agent.Mobility< 0.6)
```

overloaded nodes (Algorithm 4), and choices that require the agent to have a referral it does not have. The node then determines if the agent falls, and its severity. In the case of a severe fall the agent is sent to the Hospital; in the case of a moderate fall the agent is sent to the GP; mild falls are only recorded (they affect the agent but do not send them anywhere). The result of this filtering is passed to the agent. If there is more than one option remaining, the agent does its own filtering, starting with whether it has the energy to take that option. Of any remaining options, the agent will use its current state to choose one. The only free choice in this system is that sometimes the agent can choose between Social and Intervention. If it can, it will choose Intervention as that is considered to have a higher worth than Social, but only if the agent has sufficient resources to go to Intervention.

Agents who go to the GP are either sent to Hospital or sent Home with a referral (Algorithm 5). All agents discharged from the Hospital are referred. This is a referral for an Intervention (physiotherapy or occupational therapy). A referral is required to access the Intervention node. The Intervention node removes the referral after evaluation of the agent if the agent's mobility is over a threshold value which does not represent 'good' but 'sufficient' mobility.

As an agent loses mobility it is redirected by the nodes into Care. At this point the agent exits the system.

Capacity Setting

In the model described above, agents are referred for Intervention only after a severe fall. The follow-up care of those discharged from Hospital is the primary purpose of the Intervention. The hypothesis we test here is that with the same resources the agent population is better served if Intervention resources are available not only for those agents who have already had a severe fall, or, that prevention is better than cure.

Algorithm 6 Capacity setting Intervenor algorithm

```
IntervalHistory : list of Intervals
c := Intervention.Capacity
i := current average Interval
IntervalHistory += [i]
if Interval has increased since last week, and Interval > a week then
    Intervention.Capacity := c + 1
else if Interval has decreased since last week, and Interval < a week then
    Intervention.Capacity := max(c − 1, 0)
```

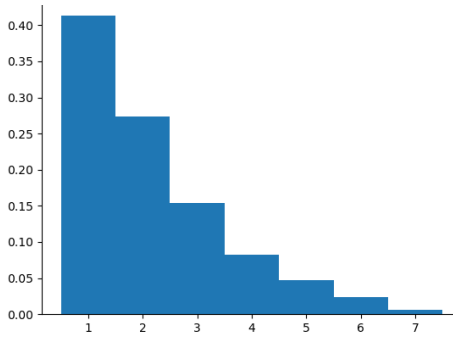

Figure 5: Histogram of frequency of capacity for the intervention node in systems with dynamic capacity, taken from all 5 runs

Before we can test this hypothesis we need to determine the amount of resource needed by the population. To do this we use a Viewer to observe the length of time between an agent leaving the Hospital with a referral and attending an Intervention. Based on the change in average time interval between referral and Intervention over the last 5 time steps we use an Intervenor to vary the capacity limit of the Intervention node (Algorithm 6): if the average is rising, we increase capacity; if the average is falling, we decrease capacity. This gives a capacity that fluctuates with the demand.

We perform 2000 time steps with 750 agents on our network with the capacity adjustment Viewer. A second program ensures that the system always has 750 active agents. It is not triggered by the patient arriving at Care, rather it watches the system and if it sees there are less than 750 agents it adds new ones at the Home node.

We perform 5 runs and collect the capacity values over time in each run. This gives us a histogram (figure 5) that tells us the adjusted capacity at any point in time. We aim to have sufficient capacity as much as possible with out assigning more resources than needed.

From the histogram we see that we can cover the demand 68% of the time with a capacity of 2; this would be a minimal capacity representing a heavily stretched provision. A capacity of 4 should cover demand 91% of the time; this would represent a good provision without wasting additional resources. For the purpose of our next set of experiments we

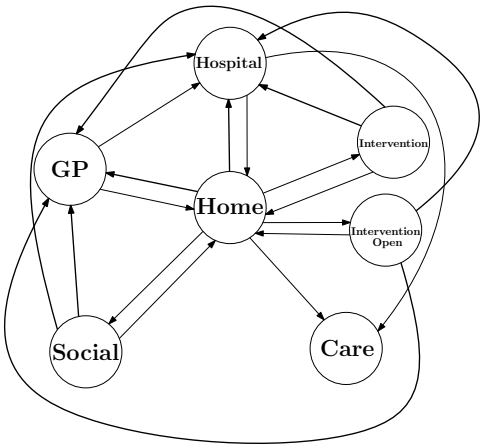

Figure 6: Network used in experimental set up with the addition of a second intervention node

use the lower capacity of 2. This allows us to test our particular hypothesis in a sparsely provisioned setting closer to reality.

Experimental Set Up

We investigate if wider access to intervention can benefit the population as a whole. To do this we take the capacity of 2 established in capacity setting and run three variant systems.

The first, 'Control', is the same as the system in the capacity setting case, but with the capacity fixed at 2 on the Intervention node.

The second, 'Open50', introduces a second intervention node, InterventionOpen (Figure 6). This operates in a similar manner to the original Intervention node, with two differences. The incoming edge no longer requires a referral, nor does it have any requirement on the status of agents using it. The capacity of the each Intervention node is 1. This gives us the same overall capacity but with some of it available to all agents.

The third, 'Dynamic', also uses the second Intervention node. It uses the same algorithm as the parameter setting to move capacity from Intervention when it is not needed to InterventionOpen, to try to keep the intervention time under 14 timesteps, and stable.

We look at two metrics for the results, both looking only at agents that have had a full life cycle in the system, from entering the system as an individual with declining mobility at risk of a fall, to when their mobility becomes too low for independent living and they move into Care. The first metric is the *system interval*: the total time from being introduced to the population until entering Care. The second is the number of *recoveries* in this time, where the agent regains enough mobility to be considered 'healthy' again, with the level of mobility at which there is negligible risk of a fall. The agent's mobility still continues to decline over time, as this model is based on decline in the elderly who are losing

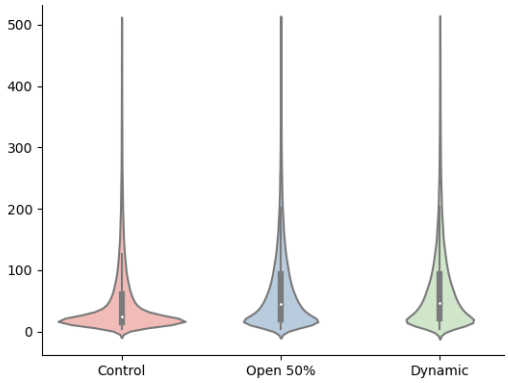

Figure 7: Distributions of System Intervals in original system, a system with open intervention available and a system with dynamic resource allocation. Median Control: 26, Median Open Intervention: 45, Median Dynamic:48. There are a few outliers that remain in the system for longer than 500 timesteps including those who remained in the system for almost the entire run, max system interval 1997

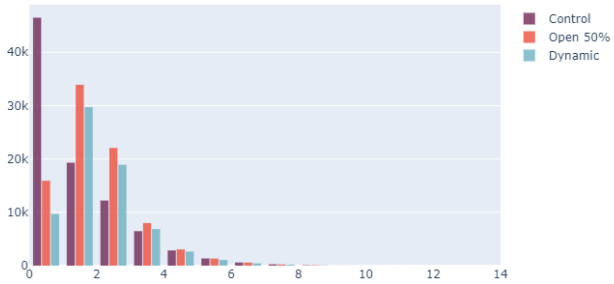

Figure 8: Distributions of Recoveries in original system, a system with open intervention available and a system with dynamic resource allocation. Median Control: 0, Median Open Intervention: 1, Median Dynamic:1.

mobility due to age such that decline cannot be completely prevented.

Results

We see an increase in the system interval for both Open50 and Dynamic, where the agents have open access to the intervention (Figure 7). The median increase is 19 time steps for Open50 and 22 time steps in the dynamic system. The p-values (Mann-Whitney U test) are effectively zero, but the effect sizes (Vargha-Delaney A value) are 0.59 for Open50 and 0.61 for Dynamic system, both of which are small effect sizes. This increase suggests there is some benefit to the population but it would be a stretch to suggest the resource re-allocation would be worth it.

For the number of recoveries (Figure 8) the p-values are again effectively zero, but the A values are 0.65 and 0.67 respectively, which are medium effect sizes. This is an indicator of possibility for improvement in the population; the change of a median number of recoveries from 0 to 1 is promising. From Figure 8 we see that the maximum num-

ber of recoveries is quiet high, with some agents having recovered 12 to 14 times. The interquartile range however changes significantly for the two new systems: the majority of agents in these systems recover at least once, whereas in the control system at least 50% never recover.

These results suggest that further investigation of these types of system is needed. These models are based on the premise that in the real world systems tend to be under pressure and over-subscribed. Here, when we take resources away from vital care we see no negative effects over the lifetime of the agents, but there will be other factors in the real world. In future work, we need to improve the accuracy of the pressure on the system based on real world numbers.

Discussion

This case study is a small initial example, intended to demonstrate the interplay between the agent model of individuals, and the network model of the environment. It could be extended to demonstrate further capabilities of the modelling approach. In ongoing work we are adding a second network to the system: a social network with nodes representing individuals including patients and groups and organisations. We are also adding carers, a second role with different behaviours, to the system, who can support patients and can be patients themselves. This social network will be a dynamic network, in which agents create and destroy social edges. There will also be Intervenors who use Viewer information to introduce additional 'group' nodes and edges representing the existence of persistent social clusters within the social network. We will also introduce change in the the current network by introducing more and varied social options that can appear and disappear over time.

Particular domain models should as far as possible reflect as much of reality as needed to test the relevant hypotheses. To do this we need domain experts to provide input to the models and hypotheses; without this interpreted domain input we cannot build realistic models. One advantage of domain modelling is that network diagrams are an accessible tool for communicating the different aspects and states in the system with those who do not have expertise in computational modelling or programming. This can help us to expand the system towards realism and towards models that can answer the questions of interest to our domain specialists.

One large gap in realism in this and many other agent based models is the discrete timestep. Reality does not happen in discrete units of time. This is why our home and hospital nodes have queues, so they can predict when the agents will move on. They are designed to work on the same basis as a discrete event simulation (DEVS), providing a joining of two well established modelling techniques (Siebers et al., 2010). This design will allow the existing system to be adjusted to work in continuous time as a DEVS implementation: the time to be spent at a node until the event that causes the agent to move on is predicted on arrival, and used to process only those agents that are changing at any given time.

These extensions could allow us to efficiently simulate continuous time transitions and structural changes in systems with a large heterogeneous population.

This sort of change modelling combined with the dynamic parameter setting we used here are two main advantages of socio-physical modelling. It allows us to develop a stable state system that reflects our data by re-balancing parameters live in the system. We can then begin to implement change in the system to see the agent reaction and the effect on regular system dynamic processes.

We can use this modelling to consider the process of change and if how the change is implemented makes a difference to the eventual steady system state.

This is a starting point for tackling the 'wicked' problems that pervade many social areas: health, agriculture, economics. Applications might include looking into the effects of small interventions on public health, testing best practice for preventing food fraud and pest spread, or even looking into the knock-on effects of a pandemic on individual spending and economic recovery for small business.

Conclusion

The NetSim meta-model is designed for, but not limited to, contextualised human systems. This is a very broad remit. There are cases where contextualisation of this sort would be unnecessary for an effective agent based model since there is limited feedback from the environment. In most complex human systems there is feedback between individual and population behaviour and the environment.

The meta-model presented here, in concert with the simulation framework software developed for this example, has the potential to be applicable across a broad range of domains. It has the ability to look not just at multiple versions of a complex system, but at what changing the system would do to the population and the individual agents, possibly unveiling perverse incentives learnt during transition or in the new system. These are visible in this model as the learnt behaviour of the agent is separated from the environment.

Acknowledgements

This work was funded by the University of Hull Cluster for the Resilience of Socio-Economic Systems.

References

Banzhaf, W., Baumgaertner, B., Beslon, G., Doursat, R., Foster, J. A., McMullin, B., de Melo, V. V., Miconi, T., Spector, L., Stepney, S., and White, R. (2016). Defining and simulating open-ended novelty: Requirements, guidelines, and challenges. *Theory in Biosciences*, 135(3):131–161.

Blythe, J. and Tregubov, A. (2019). FARM: Architecture for distributed Agent-Based social simulations. In *Massively Multi-*

Agent Systems II, volume 11422 of *LNCS*, pages 96–107. Springer.

Bonabeau, E. (2002). Agent-based modeling: methods and techniques for simulating human systems. *Proc. Natl. Acad. Sci. U. S. A.*, 99 Suppl 3:7280–7287.

Epstein, J. M. (2006). *Generative Social Science: Studies in Agent-Based Computational Modeling*. Princeton University Press.

Gamma, E., Helm, R., Johnson, R. E., and Vlissides, J. (1995). *Design Patterns: elements of reusable object-oriented software*. Addison-Wesley.

Gilbert, N. and Terna, P. (2000). How to build and use agent-based models in social science. *Mind & Society*, 1(1):57–72.

Győri, B., Gácsi, M., and Miklósi, Á. (2010). Friend or foe: Context dependent sensitivity to human behaviour in dogs. *Appl. Anim. Behav. Sci.*, 128(1):69–77.

Henscheid, Z., Middleton, D., and Bitinas, E. (2006). Pythagoras: An Agent-Based simulation environment. *The Scythe : Proceedings and Bulletin of the International Data Farming Community*, 1:40–44.

Hoverd, T. and Stepney, S. (2015). Environment orientation: a structured simulation approach for agent-based complex system. *Natural Computing*, 14(1):83–97.

Klügl, F. and Davidsson, P. (2013). AMASON: Abstract Metamodel for Agent-Based SimulatiON. In Klusch, M., Thimm, M., and Paprzycki, M., editors, *Multiagent System Technologies, MATES 2013*, volume 8076 of *LNCS*, pages 101–114. Springer.

Krasner, G. E. and Pope, S. T. (1988). A cookbook for using the Model-View Controller user interface paradigm in Smalltalk-80. *J. Object Oriented Programming*, 1(3):26–49.

Macal, C. M. and North, M. J. (2010). Tutorial on agent-based modelling and simulation. *Journal of Simulation*, 4:151–162.

MacArthur, R. H. and Wilson, E. O. (2001). *The Theory of Island Biogeography*. Princeton University Press.

Měch, R. and Prusinkiewicz, P. (1996). Visual models of plants interacting with their environment. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 397–410. ACM.

Merali, Y. (2006). Complexity and information systems: The emergent domain. *J. Inf. Technol. Impact*, 21(4):216–228.

Omicini, A., Ricci, A., and Viroli, M. (2008). Artifacts in the A&A meta-model for multi-agent systems. *Autonomous agents and multi-agent systems*, 17(3):432–456.

Palmer, J., Sorda, G., and Madlener, R. (2015). Modeling the diffusion of residential photovoltaic systems in italy: An agent-based simulation. *Technol. Forecast. Soc. Change*, 99:106–131.

Purvis, M. K., Purvis, M. A., and Frantz, C. (2014). CKSW: A Folk-Sociological Meta-Model for Agent-Based Modelling. In *Social.Path Workshop, University of Surrey*. www.ias.surrey.ac.uk/workshops/computational/index.php .

Rahman, A., Setayeshi, S., and Shamsaei, Z. M. (2007). An analysis to wealth distribution based on sugarscape model in an artificial society. *International Journal of Engineering*, 20(3):211–224.

Siebers, P. O., Macal, C. M., Garnett, J., Buxton, D., and Pidd, M. (2010). Discrete-event simulation is dead, long live agent-based simulation! *Journal of Simulation*, 4(3):204–210.

Silverman, E. (2018). Bringing ALife and complex systems science to population health research. *Artif. Life*, 24(3):220–223.

Stepney, S., Polack, F. A. C., Alden, K., Andrews, P. S., Bown, J. L., Droop, A., Greaves, R., Read, M., Sampson, A. T., Timmis, J., and Winfield, A. F. T. (2018). *Engineering Simulations as Scientific Instruments: a pattern language*. Springer.